НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2020р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки
6.050101 “ Комп’ютерні науки та інформаційні технології “

на тему Вплив коефіцієнта згладжування на вигляд інтерполяційної функції Гауса

Виконав (-ла): студент (-ка) 4 курсу, групи ТР-62

Городецький Микола Вадимович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник к.т.н., доцент Сидоренко Ю. В. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент к.т.н., ст.викл.каф. ТЕУТ та АЕС
Рачинський А.Ю. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 6.050101 Комп’ютерні науки та інформаційні технології

Спеціалізація Геометричне моделювання в інформаційних системах

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту

Городецькому Миколі Вадимовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Вплив коефіцієнта згладжування на вигляд інтерполяційної функції Гауса

керівник роботи к.т.н., доцент Сидоренко Юлія Всеволодівна
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__” __ 20__р. №_____

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи мова програмування C#, мова програмування .NET, середовище розробки Visual Studio 2020, дані всесвітньої організації по захворюванню COVID-19, методи інтерполяції, тип кроку між вузлами інтерполяції

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Аналіз згладжуючого коефіцієнта α у інтерполяції методами Гаус-функції. Вибір програмних засобів реалізації програмного рішення. Розробка програмного комплексу для аналізу згладжуючого коефіцієнта α та можливості порівняти результати при різних його значеннях.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) порівняльні рисунки при різних значеннях згладжуючого коефіцієнта α; порівняльні рисунки для різного типу кроку при інтерполяції методами Гаус-функції; архітектура програмного забезпечення.

6. Дата видачі завдання "15" листопада 2019р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	16.11.19 - 10.01.20	
2.	Вивчення та аналіз задачі	11.01.20 - 15.03.20	
3.	Розробка архітектури та загальної структури системи	16.03.20 - 04.04.20	
4.	Розробка структур окремих підсистем	05.04.20 - 15.05.20	
5.	Програмна реалізація системи	16.05.20	
6.	Оформлення пояснювальної записки	20.05.20 - 29.05.20	
7.	Захист програмного продукту	09.06.20	
8.	Передзахист		

Студент _____ Городецький М.В.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Сидоренко Ю.В.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Робота присвячена аналізу впливу коефіцієнта згладжування на вигляд інтерполяційної функції Гауса.

Пояснювальна записка складається з 102 аркушів, 34 ілюстрацій, 5 додатків та 33 літературних джерел. В першому розділі наведено методи розв’язання задач обробки масивів даних методами інтерполяції, в другому розділі описується теоретичні засоби класичних методів інтерполяції та інтерполяційної функції Гауса, в третьому розділі описується аналіз порівняльних результатів та проведено аналіз похибки із застосуванням варіативного коефіцієнта згладжування, в четвертому розділі описується вибір засобів програмної реалізації, в п’ятому розділі описується архітектура та методика роботи користувача з програмним продуктом.

Ключові слова: інтерполяція, інтерполяційна функція Гауса, похибка інтерполяції, варіативний коефіцієнт.

ANNOTATION

The aim of the work is analysis of the influence of the smoothing coefficient on the form of the Gaussian interpolation function

The research paper consists of one hundred and two sheets, thirty-four illustrations, five appendixes and thirty-three sources. The first section deals with methods of solving problems of data array processing by interpolation methods are given, the second section deals with the theoretical means of classical methods of interpolation and Gaussian interpolation function, the third section describes the analysis of comparative results is described and the error analysis is performed using a variable smoothing coefficient, the fourth section describes the choice of software implementation, the fifth describes the architecture of software.

Key words: interpolation, Gaussian interpolation function, interpolation error, variable coefficient.

ЗМІСТ

ВСТУП

Задача інтерполяції набуває особливої актуальності у наш час, оскільки розвиток науки і техніки викликає необхідність обробляти великі масиви даних.

Великі наукові інститути, фірми та компанії вкладають значні кошти у розвиток цього напрямку. Серед знаних компаній і корпорацій можна назвати Google, Yahoo! та інші. За потреби зменшення часу обробки та збільшення точності результату виникає необхідність створення апарату інтерполяційної обробки даних за рахунок аналізу проблем класичних методів, та удосконалення сучасних.

Задача отримання значень інтерполяційної функції з мінімальними похибками виникає у великій кількості сфер людської діяльності, наприклад, при геометричному моделюванні об’єктів і процесів, будівництві, побудові реальних сцен (3D-зображень), захисті конфіденційної інформації [14], у літакобудуванні, банківській сфері, робототехніці [2] [3], медицині, захисті комп’ютерної інформації [13], астрофізиці [6] та інших.

Серед численних класичних методів інтерполяції можна виокремити метод Лагранжа. На практиці доволі часто його використовують для розв’язання задачі інтерполяції. Теоретичний матеріал можна знайти в багатьох джерелах, але матеріалів щодо його застосування вкрай мало. Відомо, що поліном Лагранжа дає великі похибки при великому масиві заданих вузлів інтерполяції, оскільки залежить від кількості заданих точок. Але є й інші недоліки. Тому необхідно провести глобальний аналіз вад цього методу, щоб запропонувати можливість боротьби з цими вадами за рахунок, наприклад, застосування інших методів інтерполяції, зокрема методів, які не залежать від кількості заданих точок. Таким методом є інтерполяційний метод Гауса та його модифікації.

В даній бакалаврській роботі будуть ставитись питання розробки системи, що допоможе проаналізувати різні варіанти застосування інтерполяційної функції Гауса до задач обробки даних з урахуванням певних заданих умов.

1. ЗАДАЧА ОБРОБКИ МАСИВІВ ДАНИХ МЕТОДАМИ ІНТЕРПОЛЯЦІЇ

Швидкій обробці даних у наш час приділяється багато уваги. Оскільки існує проблема застосування класичних методів інтерполяції при обробці великих масивів даних, була поставлена задача створення системи аналізу впливу коефіцієнта згладжування на вигляд інтерполяційної функції Гауса [7].

Необхідно створити програмний продукт, який надав би можливість проаналізувати існуючі методи поліноміальної інтерполяції на прикладах стандартних функцій, надав змогу автоматично оцінювати похибку інтерполяції та змінювати крок між вузлами інтерполяції.

Програмний продукт повинен вирішувати наступні задачі:

— побудова графіка досліджуваної функції та графіка результатів методів інтерполяції: Лагранжа, Гауса звичайного та Гауса параметричного — на єдиній декартовій системі координат;
— обраховувати значення функції в заданій точці графіка;
— обирати можливі варіанти досліджуваних функцій;
— змінювати варіативний коефіцієнт α в режимі реального часу;
— створювати звіт для всіх досліджуваних функції.

Для системи необхідними вхідними даними повинні бути: досліджувані функції, методи інтерполяції, тип кроку між вузлами інтерполяції, дані по розповсюдженню захворювання COVID-19 на території більшості країн світу.

На виході система повинна надавати таку інформацію: графік інтерполяції досліджуваної функції, графіки розповсюдження захворювання COVID-19, значення варіативної змінної α за замовчуванням, звіт у форматі *DOCX*.

Потенційними користувачами системи можуть бути дослідники в галузі прикладної геометрії, магістри, аспіранти, особи, зацікавлені в аналізі великих обсягів даних.

2. МЕТОДИ РОЗВ'ЯЗАННЯ ЗАДАЧІ ІНТЕРПОЛЯЦІЇ

Класичними методами інтерполяції прийнято називати поліноми Ньютона, Лагранжа та кусково-лінійну інтерполяцію. Ці методи є одними з найпростіших, але мають багато недоліків, які зупиняють програмістів від застосування їх у стандартному вигляді. Зокрема, серед основних недоліків, можна виділити значне втрачання стійкості при великій кількості вузлів інтерполяції та неоднорідному інтервалі між вузлами. Також існує ряд Гаусових функцій, які вирішують недоліки попередніх методів, та дозволяють гнучкіше використовувати інтерполяцію для досягнення бажаних результатів.

Нижче описано вищезгадані методи інтерполяції.

2.1. Наближення функцій методами класичної інтерполяції

Кусково-лінійна інтерполяція

Для того, щоб знайти функцію *y(x)* для довільної точки *x*, маючи при цьому задану таблицею [11] функцію *f(x)*, необхідно знайти інтервал $x_{i-1} <= x <= x_i$, де $i \in [0, N)$, у який потрапляє бажана точка з аргументом *x* та підставити його в формулу (2.1).

$$y(x) = y_i - ax_i; a = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}, \quad (2.1)$$

де $i \in [0, N)$

Тобто на виході матимемо шукану точку з координатами *x* та *y*.

Метод Ньютона

Нехай у рівновіддалених точках $x_i = x_0 + i\,h$, де $i \in [0, N)$, h — крок інтерполяції, задані значення $y_i = f(x_i)$ для заданої таблицею функції $y = f(x)$.

Необхідно відшукати такий поліном $P_n(x)$ степені не вище n, що задовольняє умовам $P_n(x_i) = y_i$.

Існує два види формули для інтерполяції методом Ньютона [19]: пряма (2.2) та обернена (2.3). Першу слід використати лише в околі початкової точки x_0 інтерполяційної функції $y = f(x)$, де q достатньо мале, інакше необхідно користуватись оберненою формулою.

$$P_n(x) = y_0 + q\Delta y_0 + \frac{q(q-1)}{2!}\Delta^2 y_0 + \cdots + \frac{q(q-1)\ldots(q-n+1)}{n!}\Delta^n y_0 \qquad (2.2)$$

$$P_n(x) = y_n + q\Delta y_{n-1} + \frac{q(q+1)}{2!}\Delta^2 y_{n-2} + \cdots + \frac{q(q+1)\ldots(q+n-1)}{n!}\Delta^n y_0, \qquad (2.3)$$

де $q = \frac{x - x_0}{h}$, $\Delta^k y_i = \Delta^{k-1} y_{i+1} - \Delta^{k-1} y_i$

На виході матимемо шукану точку з координатами x та y.

Метод Лагранжа

Нехай задано поліном $P_i(x_i) = 1,\ P_i(x_j) = 0$, де $i \neq j;\ i, j \in [0, N]$. Якщо $P_i = 0$ в точках $x_1, x_2, \ldots, x_n$, окрім x_i, то його можна представити у вигляді [20]:

$$P_i(x) = C_i \cdot (x - x_1)(x - x_2) \cdot \ldots \cdot (x - x_{i-1})(x - x_{i+1}) \cdot \ldots \cdot (x - x_n), \qquad (2.4)$$

де C_i — деякий постійний коефіцієнт

Нехай $x = x_i$, $P_i(x_i) = 1$, отримаємо:

$$1 = C_i (x_i - x_1)(x_i - x_2)\ldots(x_i - x_{i-1})(x_i - x_{i+1})\ldots(x_i - x_n)$$

звідки

$$C_i = \frac{1}{(x_i - x_1)(x_i - x_2)\dots(x_i - x_{i-1})(x_i - x_{i+1})\dots(x_i - x_n)}$$

Формулу для розрахунку P_i можна отримати, підставивши C_i у рівняння (2.4):

$$P_i(x) = \frac{(x - x_1)(x - x_2)\dots(x - x_{i-1})(x - x_{i+1})\dots(x - x_n)}{(x_i - x_1)(x_i - x_2)\dots(x_i - x_{i-1})(x_i - x_{i+1})\dots(x_i - x_n)}$$

Нехай многочлен $P_n(x)$, приймає значення y_i в точках x_i, тоді такий многочлен має наступний вигляд:

$$P_n(x) = \sum_{i=0}^{n} P_i(x) y_i$$

Тоді отримуємо інтерполяційний поліном Лагранжа (2.5):

$$P_n(x) = \sum_{i=1}^{n} \frac{(x - x_1)(x - x_2)\dots(x - x_{i-1})(x - x_{i+1})\dots(x - x_n)}{(x_i - x_1)(x_i - x_2)\dots(x_i - x_{i-1})(x_i - x_{i+1})\dots(x_i - x_n)} y_i \qquad (2.5)$$

Зазначимо, що цей та інші згадані методи в цьому підрозділі мають значну похибку при збільшенні кількості точок та різкій зміні відстані між цими точками.

2.2. Наближення функцій методами Гаус-інтерполяції

У цьому підрозділі будуть описані методи інтерполяції, основані на експоненціальній залежності, що обумовлює зменшення похибки із зростанням відстані між вузлами і в деяких випадках дає перевагу над традиційними методами інтерполяції, описаними вище.

Метод інтерполяції на основі функції Гауса

Задано функцію $y_i = f(x_i)$ у вигляді таблиці точок з координатами x_i та y_i.

Запишемо інтерполяційний поліном у вигляді узагальненого многочлена $\varphi(x)$:

$$\varphi(x) = a_{00}\psi_1(x) + a_{11}\psi_2(x) + ... + a_{nn}\psi_N(x),$$

де $\psi_i(x)$ — система деяких незалежних функцій.

Нехай в даній роботі $\psi_i(x)$ — це система експоненціальних функцій. Тоді інтерполяційну функцію Гауса (далі Гаус-функція) можна записати у вигляді суми експоненціальних функцій:

$$\varphi(x) = \sum_{i=1}^{n} \psi_i(x),$$

де $\psi_i(x) = \tilde{y}_i e^{-a(x-x_i)^2}, i \epsilon [1,\ N],$

α — варіативна змінна. Було прийнято обирати цю змінну наступним чином (2.6):

$$\alpha = \frac{\pi(n-1)}{(x_{\max} - x_{\min})^2}, \qquad (2.6)$$

де x_{max}, x_{min} — максимальне та мінімальне значення аргумента x.

Геометричний зміст методу показаний на рис. 2.1.

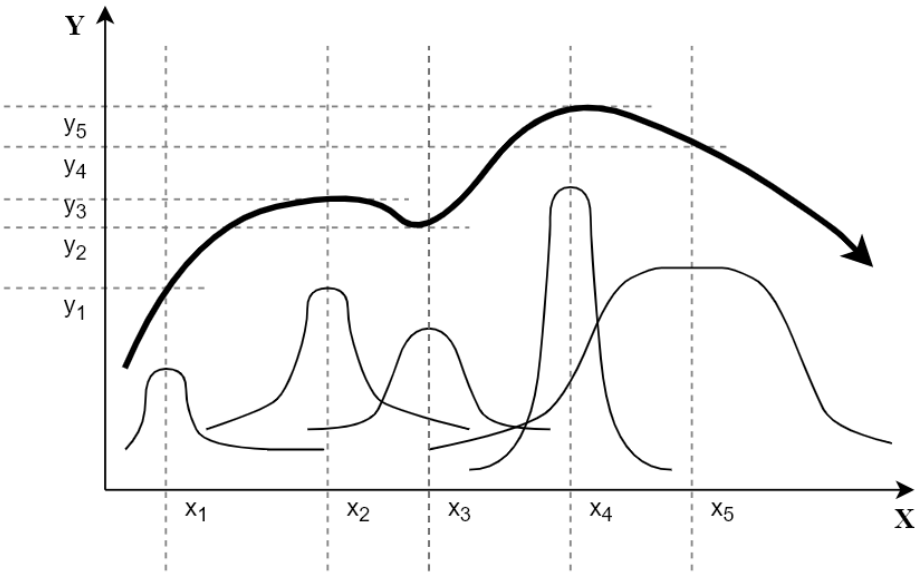

Рис. 2.1. — Геометричний зміст Гаус-функції

Для вище заданої функції $y_i = f(x_i)$ необхідно знайти базисні значення $\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_n$, на яких будуватимуться опорні функції $\psi_i(x)$.

Базисні значення знайдемо розв’язавши будь-яким чисельним методом лінійну систему (2.7). На головній діагоналі розташовані одиниці та система симетрична відносно головної діагоналі. В даній дипломній роботі для розв’язку системи було обрано метод Жордана-Гауса, так як він відповідає необхідним характеристикам щодо швидкодії та чудово оптимізує розв’язок систем з подібними характеристиками.

$$\begin{cases} \tilde{y}_1 e^{-\alpha(x_1-x_1)^2} + \tilde{y}_2 e^{-\alpha(x_1-x_2)^2} + \ldots + \tilde{y}_n e^{-\alpha(x_1-x_n)^2} = y_1, \\ \tilde{y}_1 e^{-\alpha(x_2-x_1)^2} + \tilde{y}_2 e^{-\alpha(x_2-x_2)^2} + \ldots + \tilde{y}_n e^{-\alpha(x_2-x_n)^2} = y_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \tilde{y}_1 e^{-\alpha(x_n-x_1)^2} + \tilde{y}_2 e^{-\alpha(x_n-x_2)^2} + \ldots + \tilde{y}_n e^{-\alpha(x_n-x_n)^2} = y_n. \end{cases} \quad (2.7)$$

Використаємо вектор-розв’язок цієї системи для запису інтерполяційної функції Гауса:

$$G(x) = \tilde{y}_1 e^{-\alpha(x-x_1)^2} + \tilde{y}_2 e^{-\alpha(x-x_2)^2} + \ldots + \tilde{y}_n e^{-\alpha(x-x_n)^2}.$$

Метод на основі параметричної функції Гауса

Метод Гауса добре себе показує на таких типах функцій, як: синусоїдальні, експоненціальні, параболічні, логарифмічні і т.д., коли для одного x існує тільки одне значення y [12]. Якщо необхідно задавати каркас точок для інтерполяції вручну, утворюючи при цьому петлі або замкнуті фігури (рис. 2.2), то Гаус-функції, заданої звичайним способом, не достатньо.

Для того, щоб мати можливість побудувати інтерполяцію для таких фігур, можна використати Гаус-функцію, задану параметрично. Для цього введемо параметр $t \in [1,N]$ до вище згаданої Гаус-функції. Цей параметр будемо змінювати наступним чином: для (x_1,y_1), $t = 1$, для (x_2,y_2), $t = 2$ і т.д. Для (x_n,y_n), $t = n$.

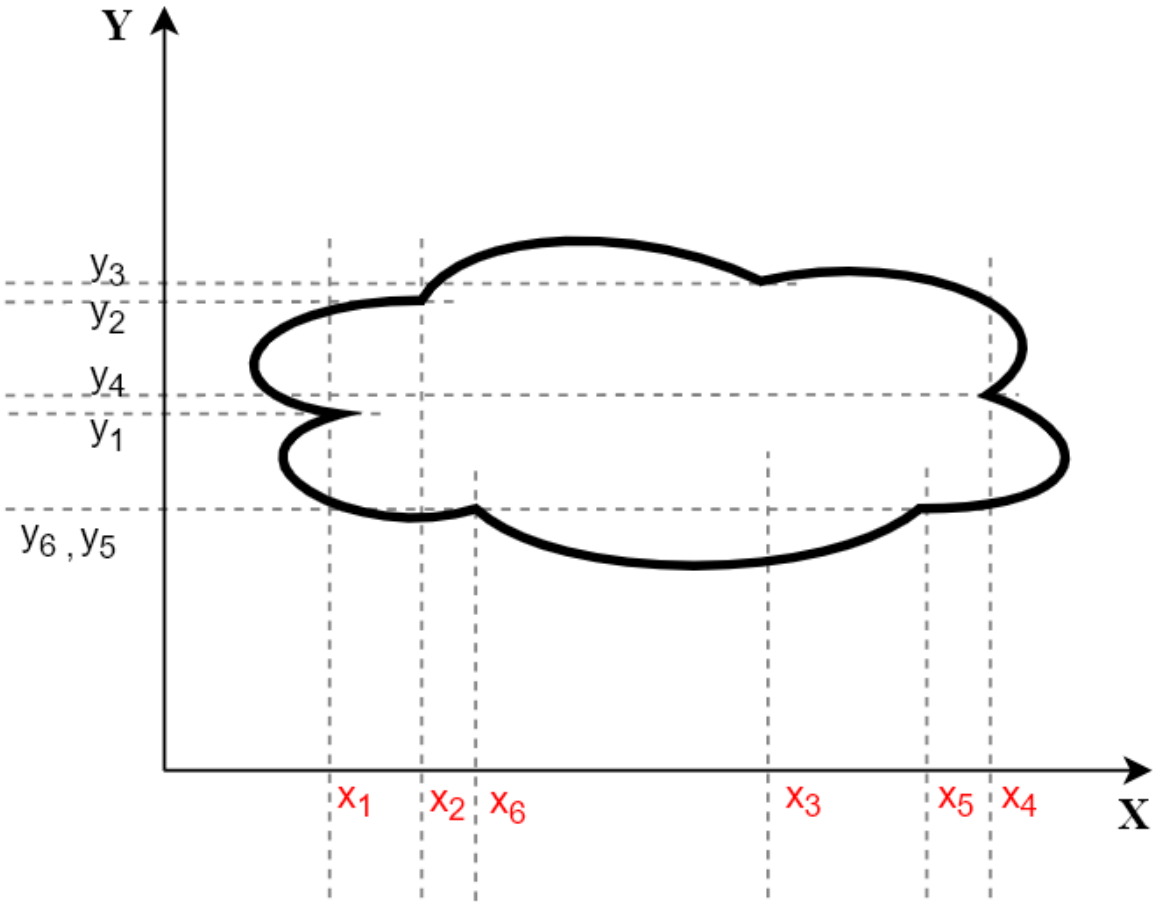

Рис. 2.2. — Застосування параметрично заданої Гаус-функції

Таким чином, кожній точці буде відповідати координата з трьох параметрів (t_i,x_i,y_i), $i \in [1, N]$ (табл. 2.1).

Таблиця 2.1. — Координати вузлів

t	**1**	**2**	**3**	**…**	**n**
x	x_1	x_2	x_3	…	x_n
y	y_1	y_2	y_3	…	y_n

Тож параметризована Гаус-функція $y=y(x)$ матиме наступний вигляд:

$$y = y\left(x_{(t)}\right) = \begin{cases} x = x(t) \\ y = y(t) \end{cases}, \tag{2.8}$$

де $x(t) = \tilde{x}_1 e^{-\alpha(t)^2} + \tilde{x}_2 e^{-\alpha(t-1)^2} + \ldots + \tilde{x}_n e^{-\alpha(t-n+1)^2}$,

$$y(t) = \tilde{y}_1 e^{-\alpha(t)^2} + \tilde{y}_2 e^{-\alpha(t-1)^2} + \ldots + \tilde{y}_n e^{-\alpha(t-n+1)^2}.$$

Для визначення базисних значень функції $x=x(t)$ $(\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n)$ необхідно розв'язати наступну систему:

$$\begin{cases} \tilde{x}_1 e^{-\alpha(t_1-t_1)^2} + \tilde{x}_2 e^{-\alpha(t_1-t_2)^2} + \ldots + \tilde{x}_n e^{-\alpha(t_1-t_n)^2} = x_1, \\ \tilde{x}_1 e^{-\alpha(t_2-t_1)^2} + \tilde{x}_2 e^{-\alpha(t_2-t_2)^2} + \ldots + \tilde{x}_n e^{-\alpha(t_2-t_n)^2} = x_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \tilde{x}_1 e^{-\alpha(t_n-t_1)^2} + \tilde{x}_2 e^{-\alpha(t_n-t_2)^2} + \ldots + \tilde{x}_n e^{-\alpha(t_n-t_n)^2} = x_n. \end{cases}$$

Для визначення базисних значень функції $y=y(t)$ $(\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_n)$, необхідно розв'язати аналогічну систему

$$\begin{cases} \tilde{y}_1 e^{-\alpha(t_1-t_1)^2} + \tilde{y}_2 e^{-\alpha(t_1-t_2)^2} + \ldots + \tilde{y}_n e^{-\alpha(t_1-t_n)^2} = y_1, \\ \tilde{y}_1 e^{-\alpha(t_2-t_1)^2} + \tilde{y}_2 e^{-\alpha(t_2-t_2)^2} + \ldots + \tilde{y}_n e^{-\alpha(t_2-t_n)^2} = y_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \tilde{y}_1 e^{-\alpha(t_n-t_1)^2} + \tilde{y}_2 e^{-\alpha(t_n-t_2)^2} + \ldots + \tilde{y}_n e^{-\alpha(t_n-t_n)^2} = y_n. \end{cases}$$

Отже, отримавши параметричне рівняння (2.8) та підставивши в нього параметр *t*, отримаємо відповідне значення *x* та *y*, за допомогою яких можна побудувати інтерполяційну криву Гауса у параметричному вигляд [9].

Метод на основі параметричної функції Гауса з набігаючим параметром *t*

Необхідність використання цього метода (далі сумарний метод) зумовлена тим, що попередній параметричний метод не враховує відстані між вузлами інтерполяції, що для деяких заданих наборів вузлів спричиняє утворення небажаних петель [15]. На рисунку 2.3. синім кольором представлена інтерполяційна крива без

урахування відстані між вузлами, рожевим — з урахуванням, червоним кольором відмічені петлі.

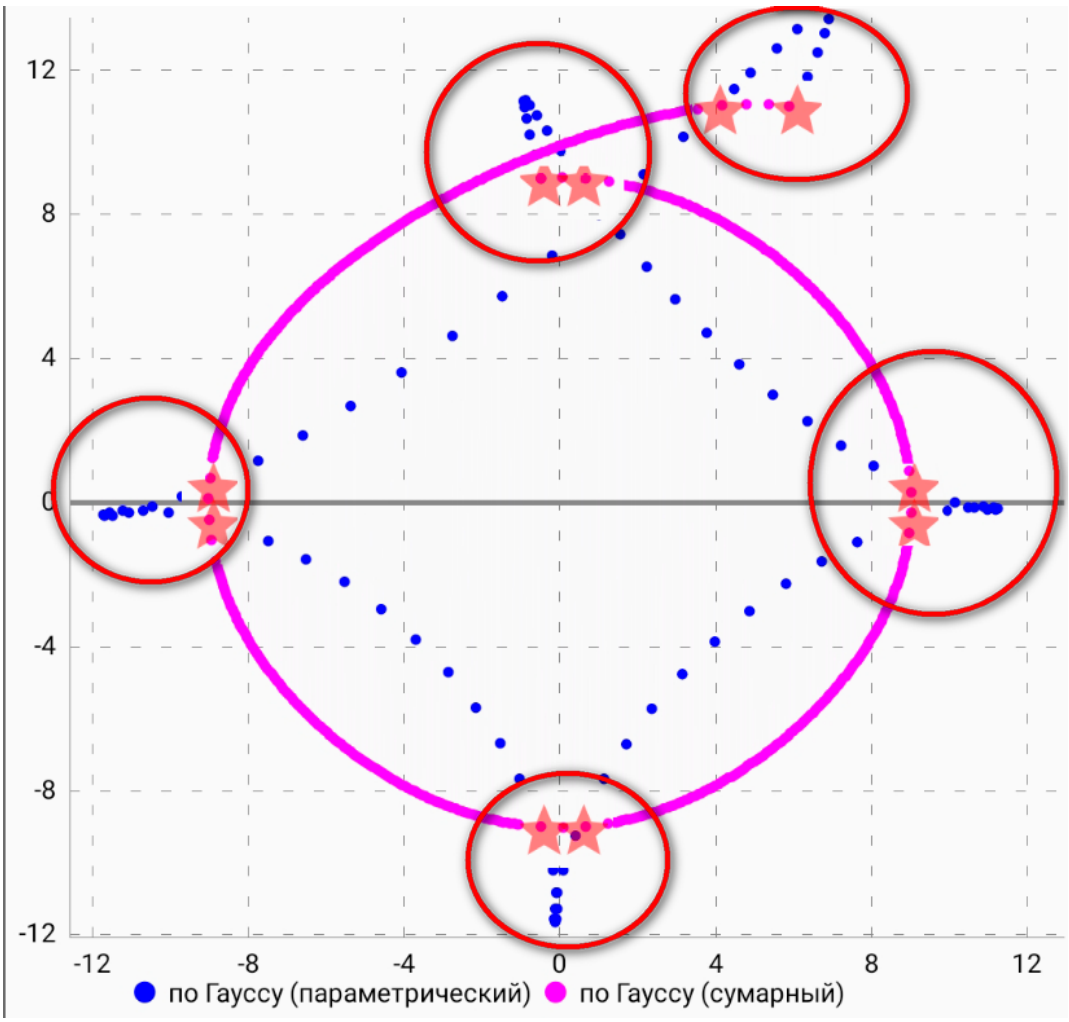

Рис. 2.3. — Різниця між параметричними методами Гауса.

Даний метод відрізняється тим, як задається параметр *t*.

Нехай $P_i(x_i,y_i)$ — це *i*-тий інтерполяційний вузол (від лат. *point*, — точка, далі просто P_i). Тоді останній вузол це $Pn(x_n,y_n)$. Задамо параметр *t* наступним чином:

$$t_i = t_{i-1} + distance(P_{i-1}, P_i),$$

де $distance(P_{i-1}, P_i)$ — це відстань між двома точками (вузлами інтерполяції),

$i \in [1, N]$, при $i = 1$, $t = 0$

Отже, кожній точці буде відповідати координата з трьох параметрів $P_i(t_i,x_i,y_i)$, $i \in [1, N]$ (табл. 2.2).

Таблиця 2.2. — Координати вузлів

$\mathbf{t_i}$	$\mathbf{t_1}$	$\mathbf{t_2}$	$\mathbf{t_3}$	**…**	$\mathbf{t_n}$
$\mathbf{t_i}$	0	$t_1 + dist(P_1,P_2)$	$t_2 + dist(P_2,P_3)$	…	$t_{n-1} + dist(P_{n-1},P_n)$
$\mathbf{P_i(x_i,y_i)}$	$P_1(x_1,y_1)$	$P_2(x_2,y_2)$	$P_3(x_3,y_3)$	…	$P_n(x_n,y_n)$

Як підсумок, існує три види інтерполяції на основі Гаус-функції, які можна використовувати окремо, або разом, дивлячись від постановки задачі [8].

3. АНАЛІЗ РЕЗУЛЬТАТІВ ПОРІВНЯННЯ ІНТЕРПОЛЯЦІЙНИХ ФУНКЦІЙ

У цьому розділі будуть наведені результати роботи деяких алгоритмів інтерполяції. Спочатку — результати інтерполяції елементарних алгебричних функцій класичним методом Лагранжа та трьома видами інтерполяційної функції Гауса з невеликою кількістю вузлів інтерполяції та з постійним кроком. Далі проведено тестування цих самих методів зі змінним кроком інтерполяції. Крім того, наведені результати інтерполяції при зміні варіативного параметра α, зроблено висновки щодо доцільності використання зміни цього параметра для різних функцій.

У розділі також розглядається інтерполяція експериментальних даних за допомогою інтерполяційної функції Гауса на прикладі розповсюдження захворювання COVID-19 на території України, зроблено висновки щодо застосування різних значень варіативного параметру для інтерполяції експериментальних даних з мінімальною похибкою.

3.1. Тестування роботи алгоритму інтерполяційної функції Гауса на елементарних алгебричних функціях

Інтерполяція з постійним кроком

На практиці доволі часто для загущення точкових каркасів використовують поліном Лагранжа. Крім випадків, коли відомі тільки дискретні значення функції, і інших шляхів, окрім інтерполяції, не існує навіть теоретично, ще виникає ситуація, коли функція задана аналітично, але дуже складно, тобто процес безпосереднього обчислення значень функцій інколи є занадто складним [29]. Тому при математичному табулюванні, як правило, функцію обраховують у невеликій кількості вузлів, а потім загущують до необхідної кількості.

Відомо, що використання поліному Лагранжа може призводити до осциляцій при великій кількості вузлів інтерполяції, оскільки степінь поліному Лагранжа дорівнює кількості заданих точок, а чим вища ступінь інтерполюючого поліному, тим більші похибки при інтерполяції. Поліном Лагранжа має й інші недоліки. Наприклад, відомо, що на виникнення небажаних осциляцій впливає різка зміна кривизни функції. Також на похибку обчислення може вплинути нерівномірний крок інтерполяції. Цей недолік було вирішено перевірити за допомогою комп'ютерного експерименту та запропонувати інші методи інтерполяції, за допомогою яких цього недоліку можна уникнути. Таким методом є інтерполяційний метод Гауса та його модифікації.

У результаті роботи було проведено комп'ютерний аналіз чотирьох методів, а саме: класичного методу Лагранжа, звичайного методу Гауса, параметричного та сумарного методів Гауса. Дослідження було проведено для різних елементарних алгебричних функцій (табл. 3.1).

Таблиця 3.1. — Досліджувані елементарні функції

$y = x^2$	$y = e^x$	$y = \sinh(x)$
$y = \frac{1}{x}$	$y = 1,3^x$	$y = \operatorname{arcsinh}(x)$
$y = \sqrt{x}$	$y = \sin(x)$	$y = \operatorname{csch}(x)$
$y = \sqrt[3]{x}$	$y = \arcsin(x)$	$y = \operatorname{sech}(x)$
$y = \lg(x)$	$y = \operatorname{arctg}(x)$	$y = \operatorname{arcsech}(x)$

Результати аналізу виявились такими.

Якщо кількість точок інтерполяції невелика (наприклад, 6 вузлів) та крок обрано постійним, то поліном Лагранжа для більшості випадків має найменшу похибку з обраних методів. На рисунках 3.1-3.3 для прикладу наведено наступні функції: $y = \sqrt{x}$, $y = \sin(x)$, $y = \operatorname{arctg}(x)$

$$y = \sqrt{x}$$

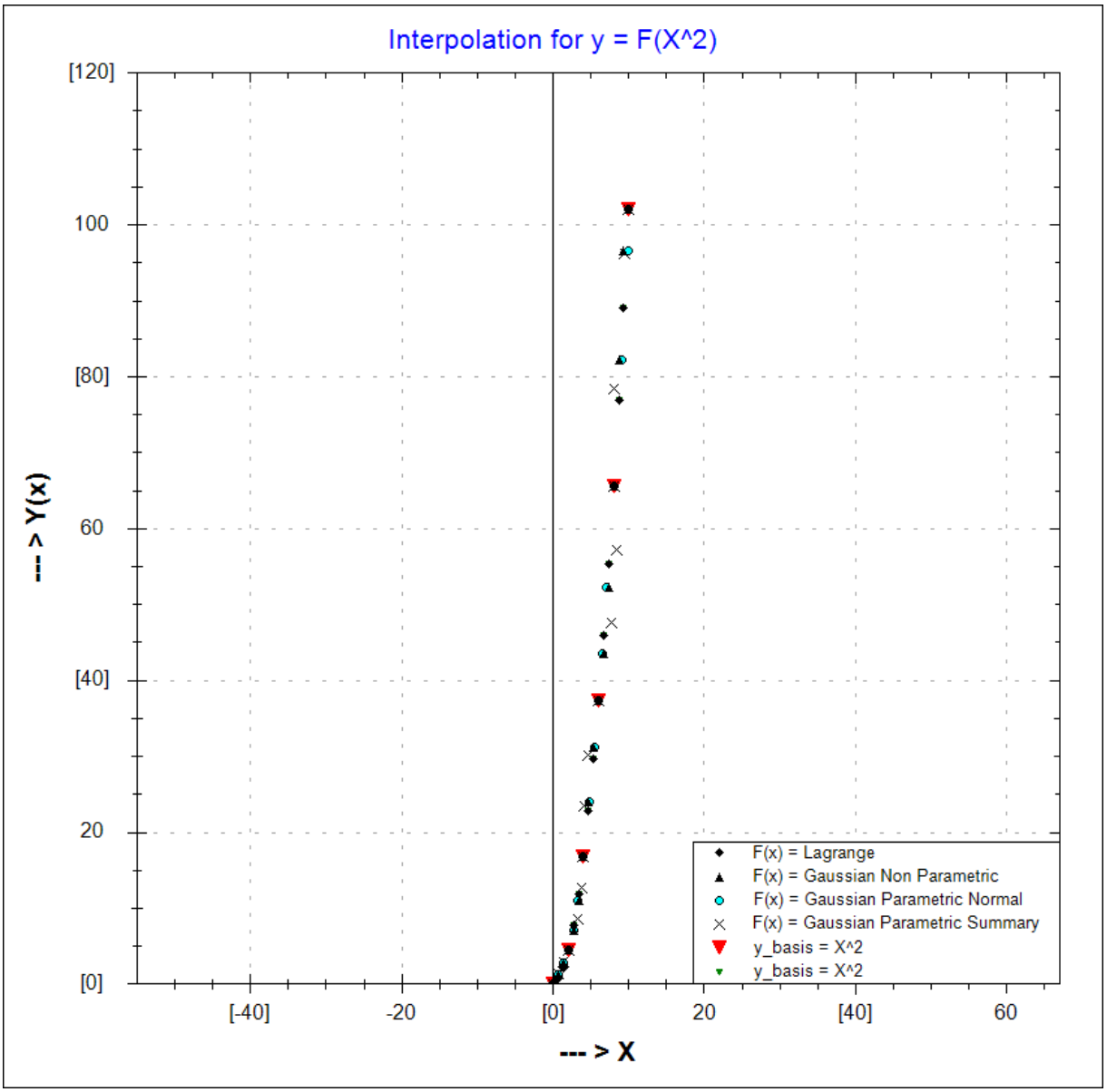

Рис. 3.1. — Інтерполяція функції $y = \sqrt{x}$ з постійним кроком

X є [0,1;10,1]. Кількість точок = 6
Крок = [1 = 0,17 2 = 0,25 3 = 0,33 4 = 0,42 5 = 0,50 6 = 0,58]

Оцінка похибки алгоритмів:

Lagrange	0,0000000000000001
Gaussian Non Parametric	0,0010235605135752
Gaussian Parametric Normal	0,0010235605135752
Gaussian Parametric Summary	0,0006071668917022

$$y = \sin(x)$$

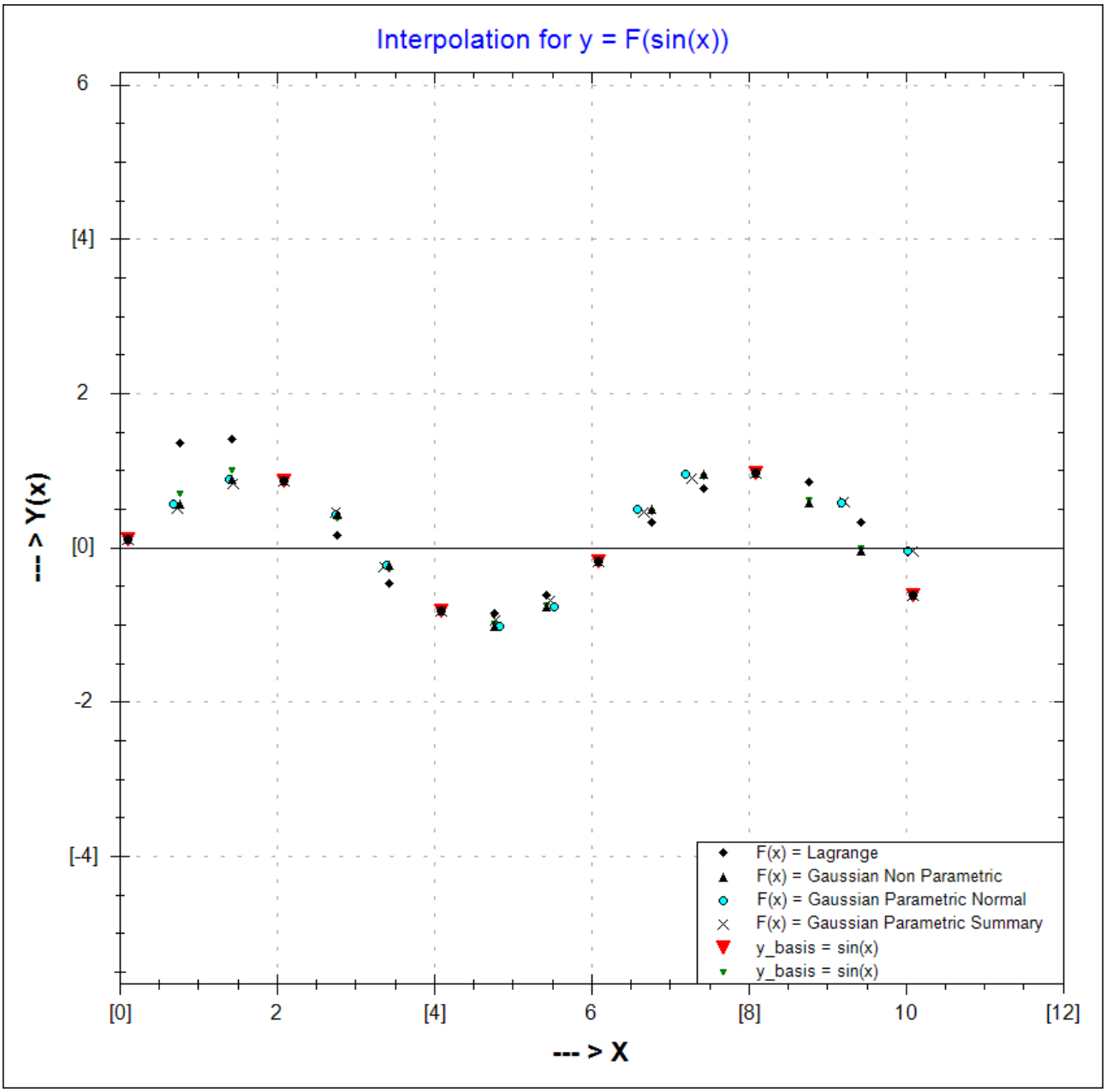

Рис. 3.2. — Інтерполяція функції $y = \sin(x)$ з постійним кроком

X є [0,1;10,1]. Кількість точок = 6
Крок = [1 = 0,17 2 = 0,25 3 = 0,33 4 = 0,42 5 = 0,50 6 = 0,58]

Оцінка похибки алгоритмів:

Lagrange	0,0933303205033280
Gaussian Non Parametric	0,0043136318410855
Gaussian Parametric Normal	0,0043136318410855
Gaussian Parametric Summary	0,0080146722153622

$$y = \text{arctg}(x)$$

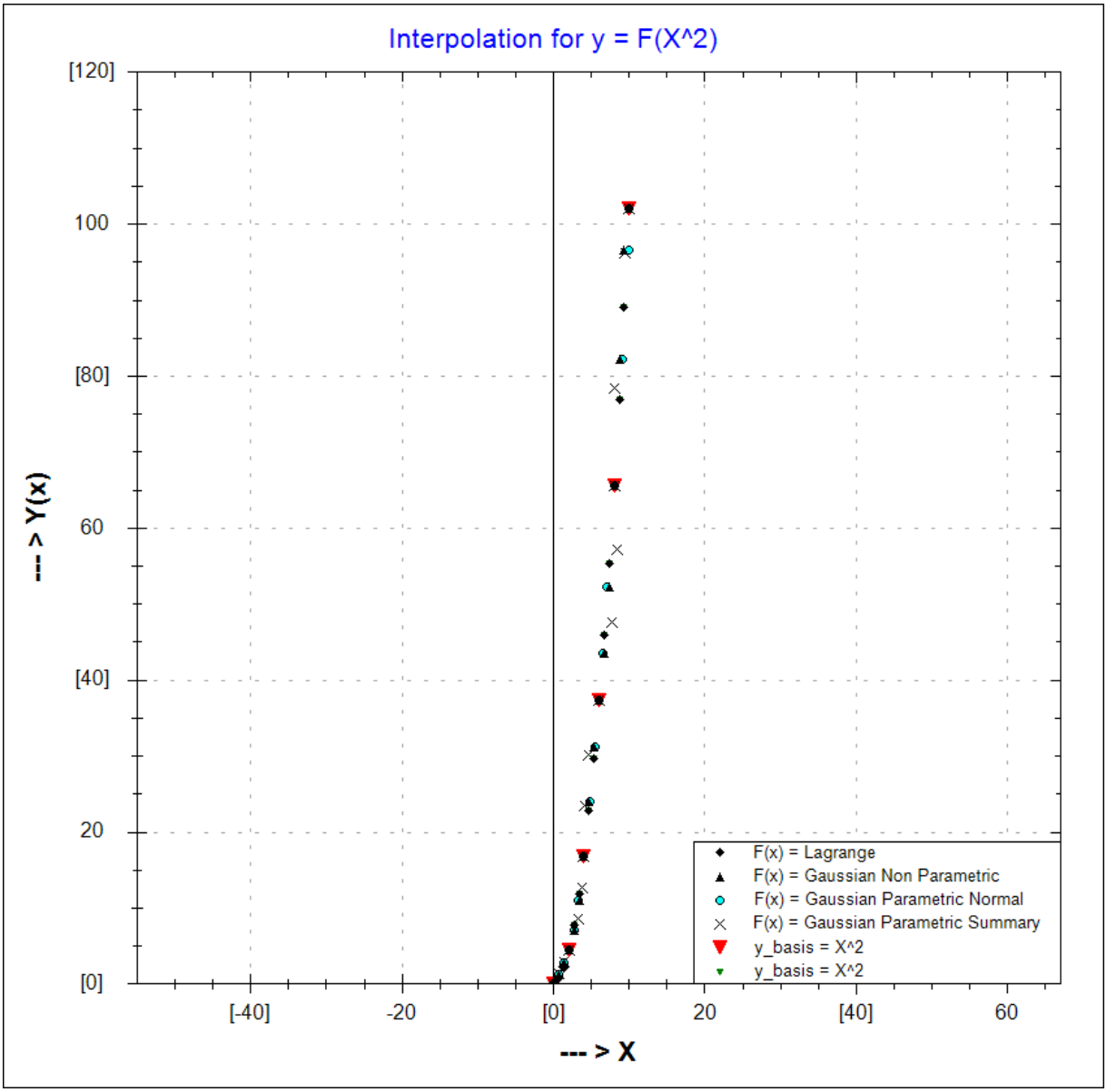

Рис. 3.3. — Інтерполяція функції $y = \text{arctg}(x)$ з постійним кроком

X є [0,1;10,1]. Кількість точок = 6
Крок = [1 = 0,17 2 = 0,25 3 = 0,33 4 = 0,42 5 = 0,50 6 = 0,58]

Оцінка похибки алгоритмів:

Lagrange	0,0001141662390737
Gaussian Non Parametric	0,0032826005354173
Gaussian Parametric Normal	0,0032826005354173
Gaussian Parametric Summary	0,0032201113194650

Інтерполяція зі змінним кроком

У випадку постійного кроку інтерполяції і при невеликій кількості вузлів класичні методи інтерполяції мають невелику похибку, але у випадку, коли крок змінний, поліном Лагранжа використовувати недоцільно, так як наявність осциляцій може нівелювати результат обчислень [25]. Для того, щоб перевірити це твердження, було проведено ряд експериментів, у яких для інтерполяції елементарних математичних функцій було введено функціональну залежність кроку, а саме x змінювався таким чином, що кожен наступний крок вдвічі більший за попередній. Розглянемо це на прикладі чотирьох точок (рис. 3.4.).

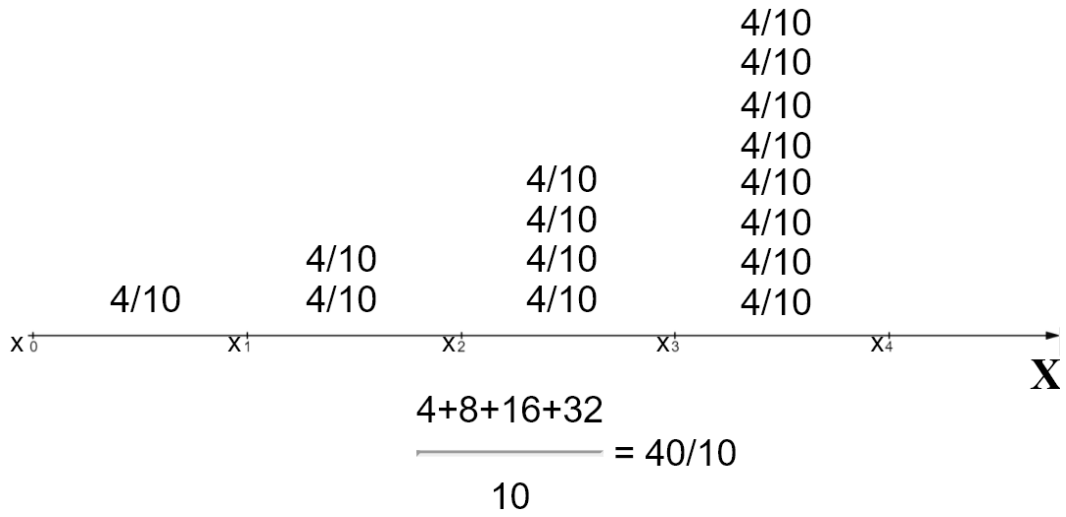

Рис. 3.4. — Розподіл кроків для чотирьох вузлів

З рисунку 3.4 можна вивести формулу, яка дозволить розрахувати крок для будь-якої точки інтерполяції, що є зручним при програмуванні алгоритму.

$$step_i = \sum_{0}^{i} \frac{x_{\max} - x_{\min}}{\sum_{h=1}^{N} h} \qquad (3.1)$$

де $step_i$ — крок інтерполяції, $i \in [1, N]$,

N — кількість точок інтерполяції

x_{max}, x_{min} — максимальне та мінімальне значення аргумента x.

Для порівняння і усунення великих похибок, інтерполяцію було проведено також за допомогою трьох варіантів інтерполяційної функції Гауса: звичайної, параметричної та сумарної.

Інтерполяційна Гаус-функція будується за допомогою функції розподілу нормального закону і є сумою опорних функцій (Гаусіан). Функція Гауса має широкий спектр використання, до якого можна віднести такі як статистика, математична фізика, теорія хвиль, обчислювальна хімія, квантова теорія поля, оптичні, мікрохвильові системи і лазери, системи комп'ютерного зору та обробки зображень, обробка сигналів, геостатистика , робототехніка, та інші. Саме тому використання для інтерполяції функцій Гауса вбачається актуальним для багатьох сфер життєдіяльності. Для прикладу наведено інтерполяцію наступних функцій: $y=\frac{1}{x}$, $y=\sqrt{x}$, $y=\sqrt[3]{x}$, $y=\lg(x)$, $y=\arcsin(x)$, $y=\operatorname{arctg}(x)$, $y=\operatorname{arcsinh}(x)$, $y=\operatorname{csch}(x)$, $y=\operatorname{sech}(x)$, $y=\operatorname{arcsech}(x)$. На рисунках 3.5-3.13 наведено результати інтерполяції та вказані похибки кожного з обраних методів.

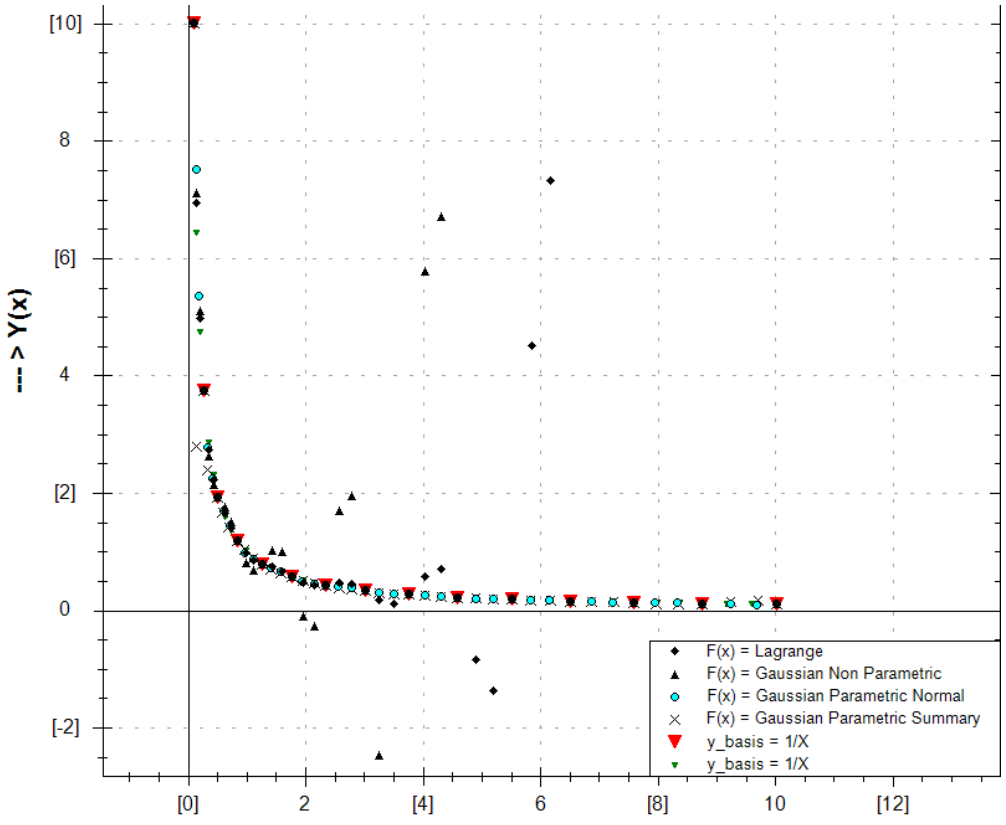

Рис. 3.5. — Інтерполяція функції $\boldsymbol{y=\frac{1}{x}}$ зі змінним кроком

X є [0,1;10,1]. Кількість точок = 15
Крок = [1 = 0,17 2 = 0,25 3 = 0,33 4 = 0,42 5 = 0,50 6 = 0,58 7 = 0,67 8 = 0,75 9 = 0,83 10 = 0,92 11 = 1,00 12 = 1,08 13 = 1,17 14 = 1,25 15 = 1,33]

Оцінка похибки алгоритмів:

Lagrange	23097,1105536787000000
Gaussian Non Parametric	0,6906856003893830
Gaussian Parametric Normal	0,0005675315297795
Gaussian Parametric Summary	0,0343563808647388

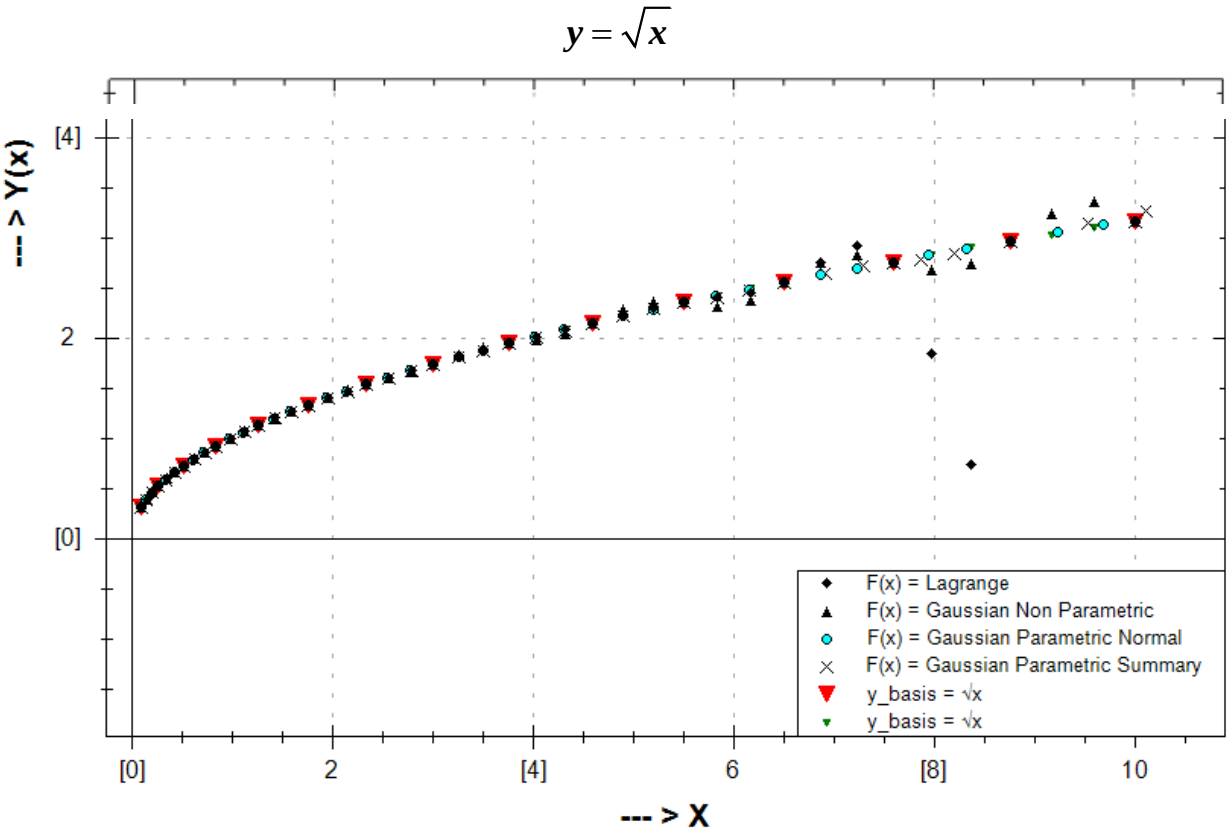

Рис. 3.6. — Інтерполяція функції $y = \sqrt{x}$ зі змінним кроком

X є [0,1;10,1]. Кількість точок = 15
Крок = [1 = 0,17 2 = 0,25 3 = 0,33 4 = 0,42 5 = 0,50 6 = 0,58 7 = 0,67 8 = 0,75 9 = 0,83 10 = 0,92 11 = 1,00 12 = 1,08 13 = 1,17 14 = 1,25 15 = 1,33]

Оцінка похибки алгоритмів:

Lagrange	5,8568776518629900
Gaussian Non Parametric	0,0008124876223612
Gaussian Parametric Normal	0,0000061058093832
Gaussian Parametric Summary	0,0001628668864064

$$y = \sqrt[3]{x}$$

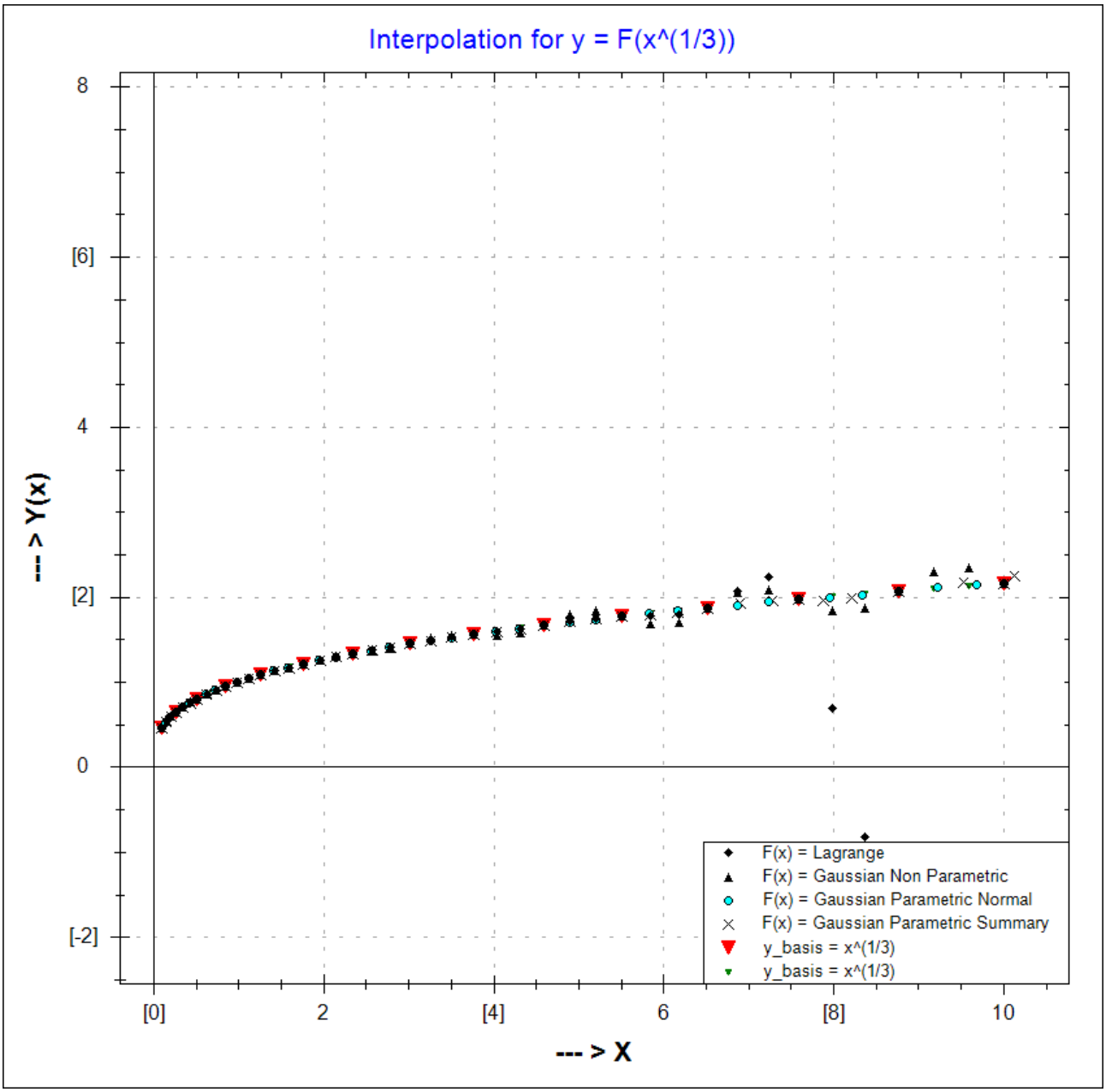

Рис. 3.7. — Інтерполяція функції $y = \sqrt[3]{x}$ зі змінним кроком

X є [0,1;10,1]. Кількість точок = 15
Крок = [1 = 0,17 2 = 0,25 3 = 0,33 4 = 0,42 5 = 0,50 6 = 0,58 7 = 0,67 8 = 0,75 9 = 0,83 10 = 0,92 11 = 1,00 12 = 1,08 13 = 1,17 14 = 1,25 15 = 1,33]

Оцінка похибки алгоритмів:

Lagrange	22,2221326691805000
Gaussian Non Parametric	0,0018489828101591
Gaussian Parametric Normal	0,0000065220588461
Gaussian Parametric Summary	0,0001953302217972

$$y = \lg(x)$$

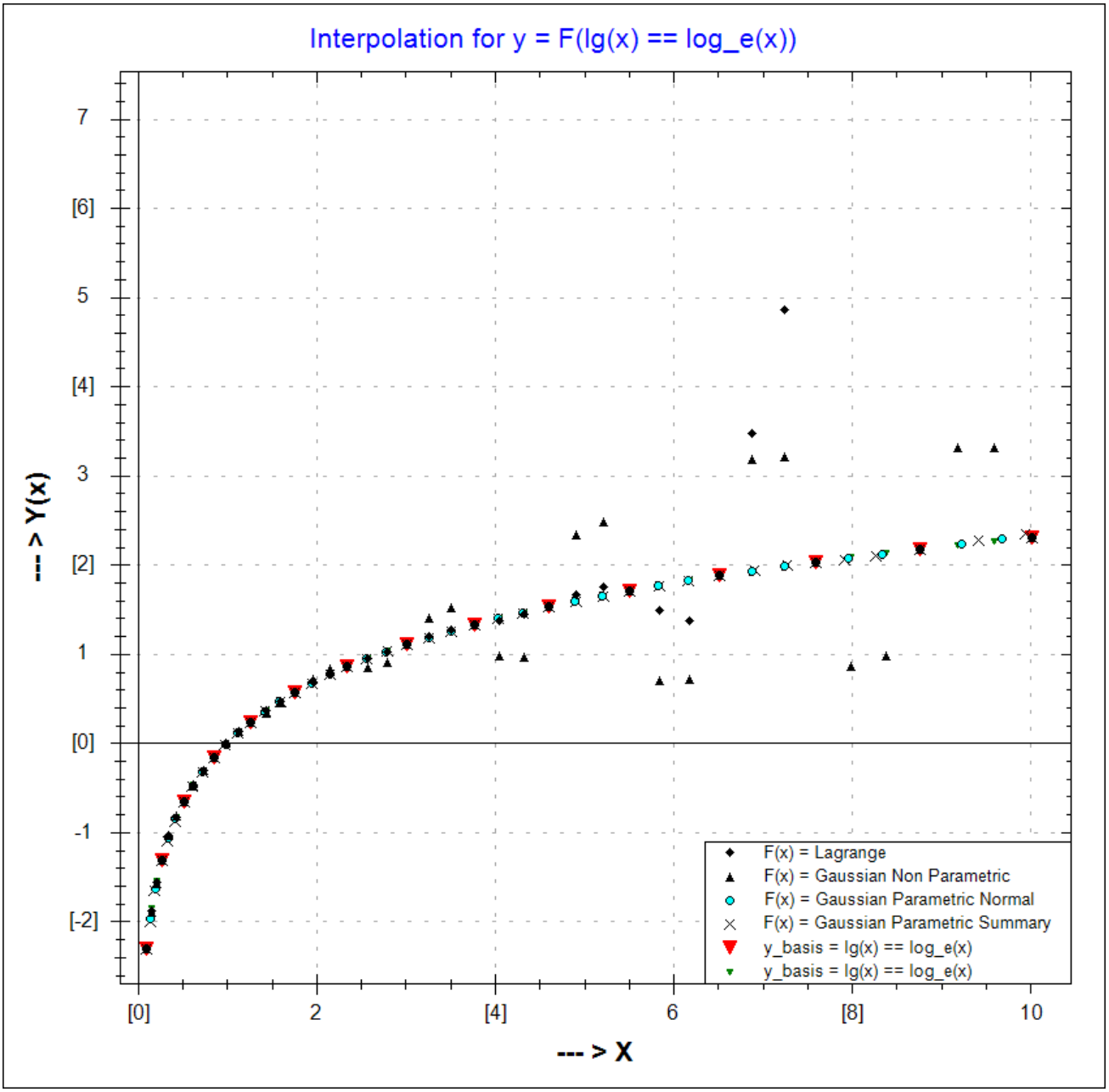

Рис. 3.8. — Інтерполяція функції $y = \lg(x)$ зі змінним кроком

X є [0,1;10,1]. Кількість точок = 15
Крок = [1 = 0,17 2 = 0,25 3 = 0,33 4 = 0,42 5 = 0,50 6 = 0,58 7 = 0,67 8 = 0,75 9 = 0,83 10 = 0,92 11 = 1,00 12 = 1,08 13 = 1,17 14 = 1,25 15 = 1,33]

Оцінка похибки алгоритмів:

Lagrange	1788,0239165818600000
Gaussian Non Parametric	0,0828547075307030
Gaussian Parametric Normal	0,0001405535256958
Gaussian Parametric Summary	0,0002856116228337

$$y = \arcsin(x)$$

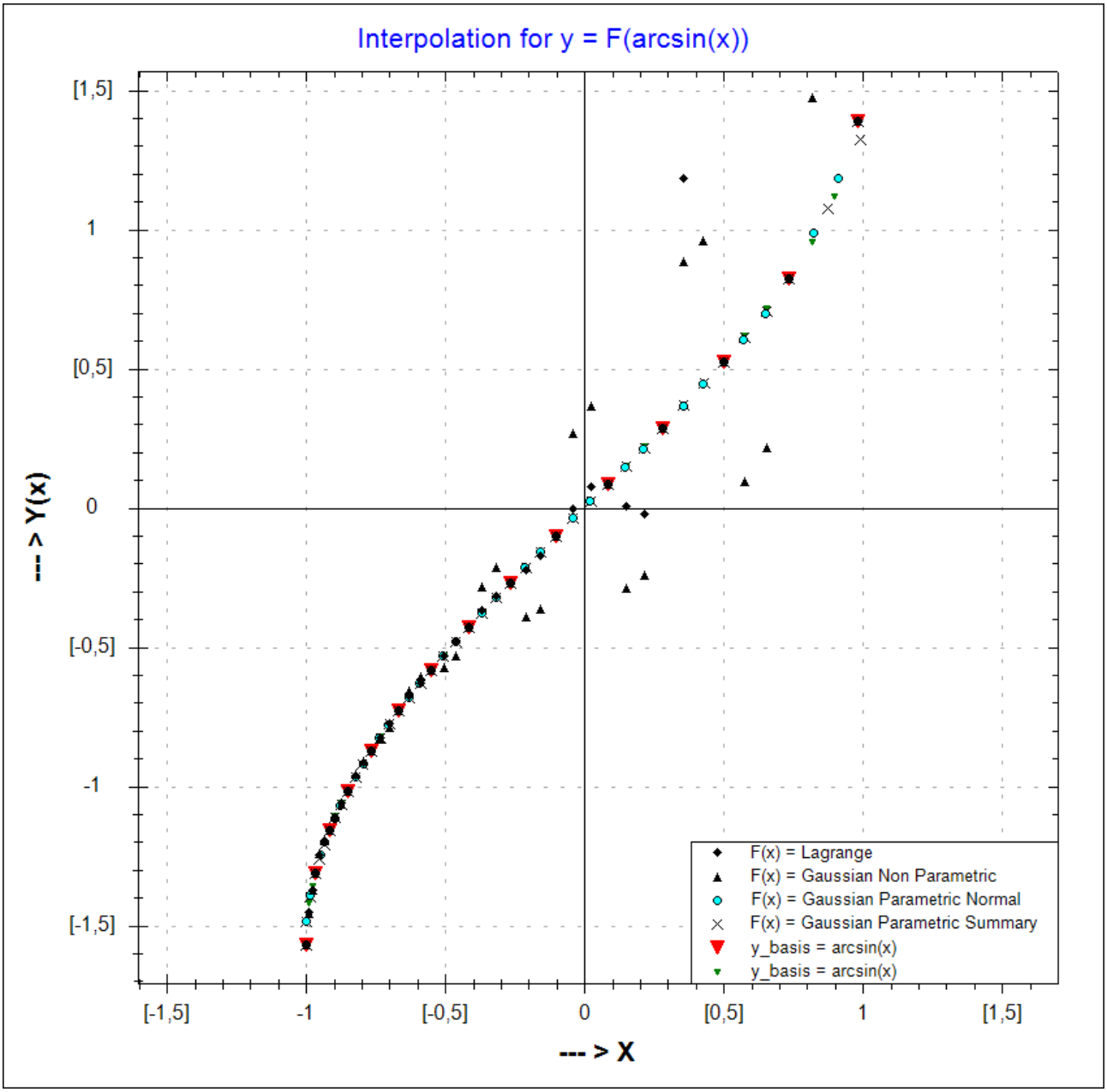

Рис. 3.9. — Інтерполяція функції $y = \arcsin(x)$ зі змінним кроком

X є [-1;1]. Кількість точок = 15
Крок = [1 = 0,03 2 = 0,05 3 = 0,07 4 = 0,08 5 = 0,10 6 = 0,12 7 = 0,13 8 = 0,15 9 = 0,17 10 = 0,18 11 = 0,20 12 = 0,22 13 = 0,23 14 = 0,25 15 = 0,27]

Оцінка похибки алгоритмів:

Lagrange	1376,4662709950100000
Gaussian Non Parametric	0,0434722329694791
Gaussian Parametric Normal	0,0002130220846603
Gaussian Parametric Summary	0,0011606748896557

$$y = \text{arctg}(x)$$

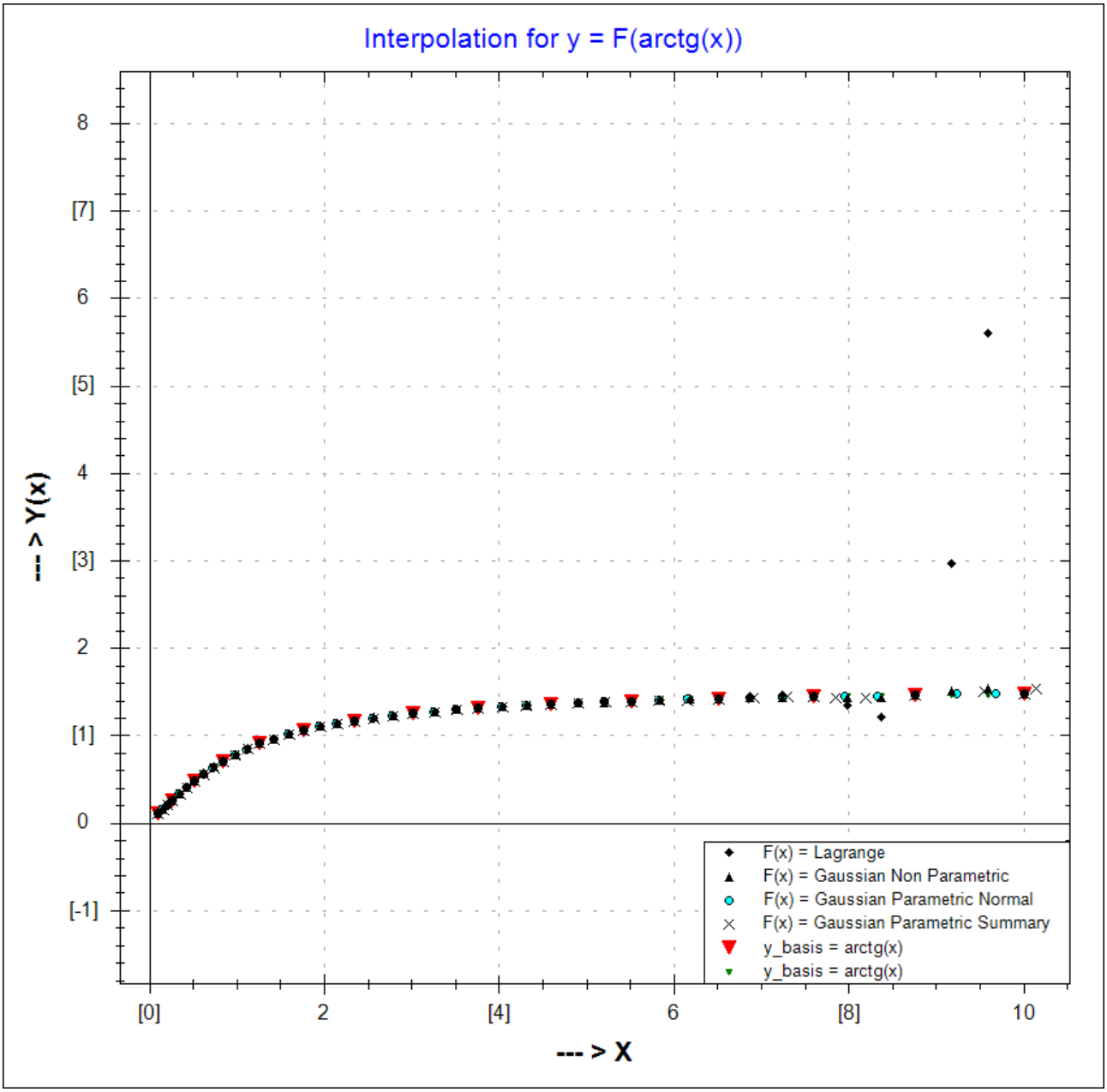

Рис. 3.10. — Інтерполяція функції $y = \text{arctg}(x)$ зі змінним кроком

X є [0,1;10,1]. Кількість точок = 15
Крок = [1 = 0,17 2 = 0,25 3 = 0,33 4 = 0,42 5 = 0,50 6 = 0,58 7 = 0,67 8 = 0,75 9 = 0,83 10 = 0,92 11 = 1,00 12 = 1,08 13 = 1,17 14 = 1,25 15 = 1,33]

Оцінка похибки алгоритмів:

Lagrange	0,3193392405453560
Gaussian Non Parametric	0,0001213970640791
Gaussian Parametric Normal	0,0000106432102438
Gaussian Parametric Summary	0,0001471943595850

$$y = \mathbf{arcsinh}(x)$$

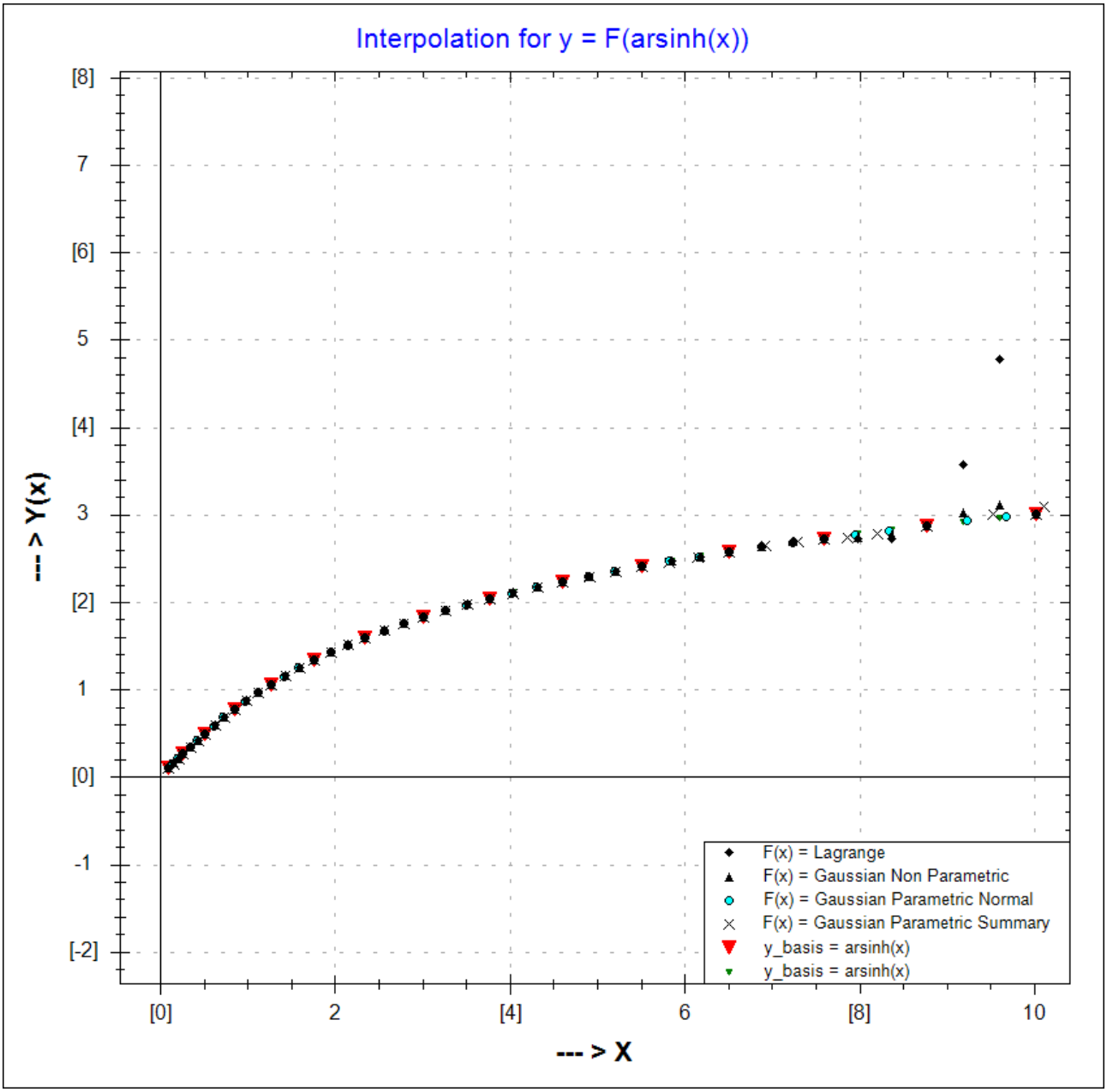

Рис. 3.11. — Інтерполяція функції $y = \text{arcsinh}(x)$ зі змінним кроком

X є [0,1;10,1]. Кількість точок = 15
Крок = [1 = 0,17 2 = 0,25 3 = 0,33 4 = 0,42 5 = 0,50 6 = 0,58 7 = 0,67 8 = 0,75 9 = 0,83 10 = 0,92 11 = 1,00 12 = 1,08 13 = 1,17 14 = 1,25 15 = 1,33]

Оцінка похибки алгоритмів:

Lagrange	0,0148152791519870
Gaussian Non Parametric	0,0001608723656775
Gaussian Parametric Normal	0,0000062659163775
Gaussian Parametric Summary	0,0001302251149526

$$y = \text{csch}(x)$$

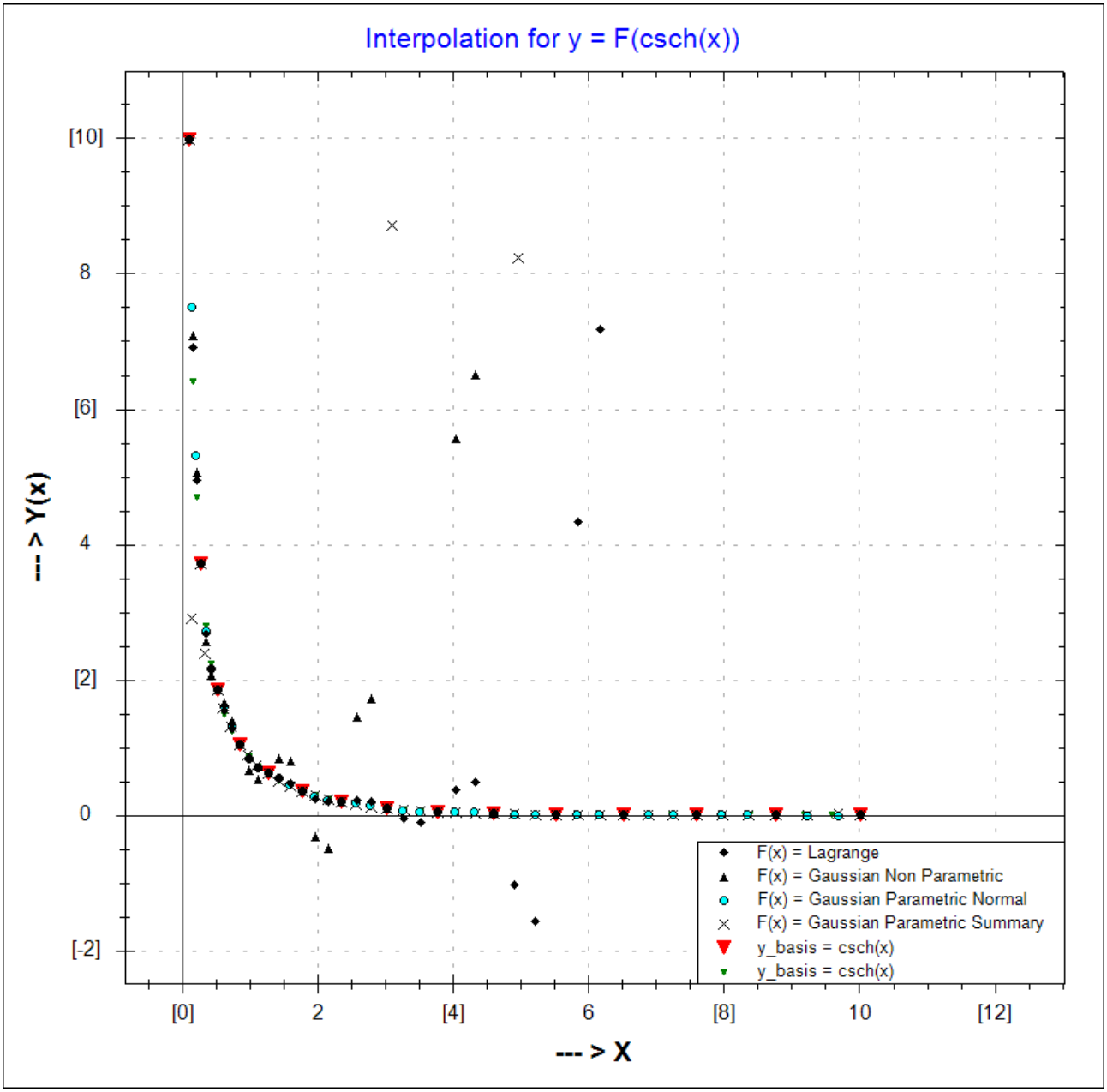

Рис. 3.12. — Інтерполяція функції $y = \text{csch}(x)$ зі змінним кроком

X є [0,1;10,1]. Кількість точок = 15
Крок = [1 = 0,17 2 = 0,25 3 = 0,33 4 = 0,42 5 = 0,50 6 = 0,58 7 = 0,67 8 = 0,75 9 = 0,83 10 = 0,92 11 = 1,00 12 = 1,08 13 = 1,17 14 = 1,25 15 = 1,33]

Оцінка похибки алгоритмів:

Lagrange	23173,978239785900
Gaussian Non Parametric	0,6931351349856580
Gaussian Parametric Normal	0,0005713235132022
Gaussian Parametric Summary	0,0064158093907819

$$y = \mathbf{sech}(x)$$

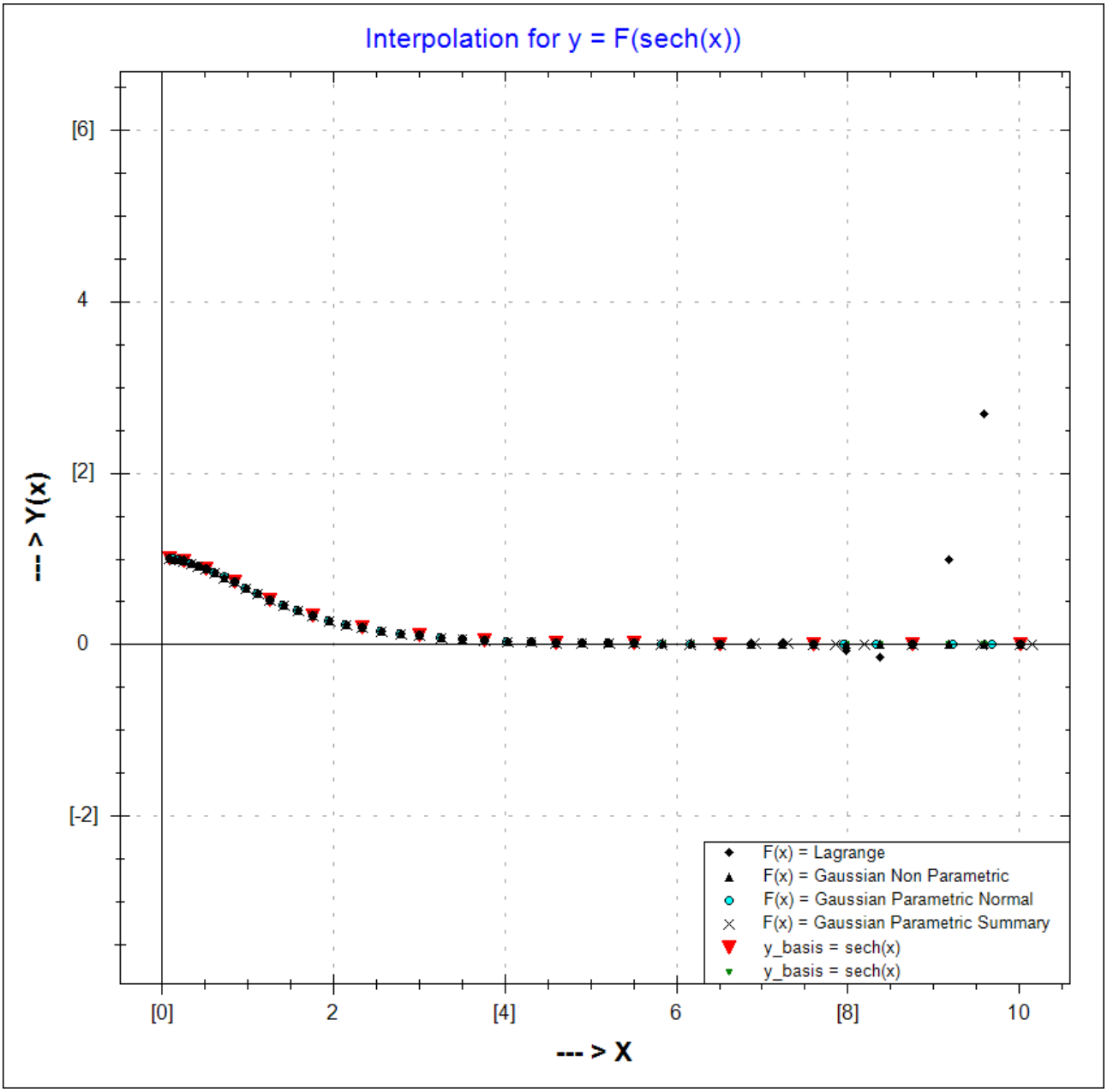

Рис. 3.13. — Інтерполяція функції $y = \text{sech}(x)$ зі змінним кроком

X є [0,1;10,1]. Кількість точок = 15
Крок = [1 = 0,17 2 = 0,25 3 = 0,33 4 = 0,42 5 = 0,50 6 = 0,58 7 = 0,67 8 = 0,75 9 = 0,83 10 = 0,92 11 = 1,00 12 = 1,08 13 = 1,17 14 = 1,25 15 = 1,33]

Оцінка похибки алгоритмів:

Lagrange	0,2970487925835390
Gaussian Non Parametric	0,0000040983877857
Gaussian Parametric Normal	0,0000091738555589
Gaussian Parametric Summary	0,0000104636214925

$$y = \mathbf{arcsech}(x)$$

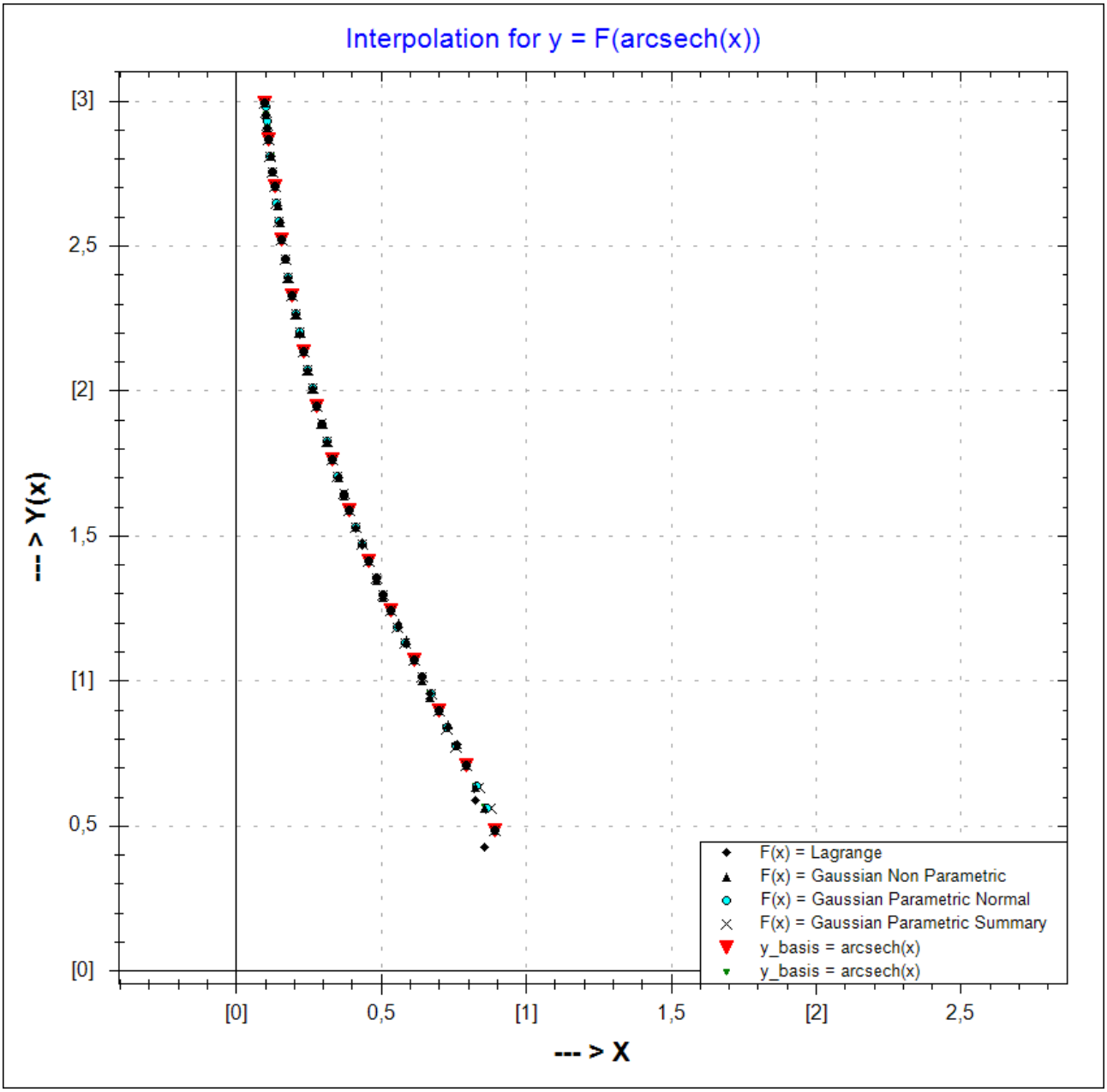

Рис. 3.14. — Інтерполяція функції $y = \text{arcsech}(x)$ зі змінним кроком

X є [0,1;0,9]. Кількість точок = 15
Крок = [1 = 0,01 2 = 0,02 3 = 0,03 4 = 0,03 5 = 0,04 6 = 0,05 7 = 0,05 8 = 0,06 9 = 0,07 10 = 0,07 11 = 0,08 12 = 0,09 13 = 0,09 14 = 0,10 15 = 0,11]

Оцінка похибки алгоритмів:

Lagrange	0,0000840554205991
Gaussian Non Parametric	0,0000051249177287
Gaussian Parametric Normal	0,0000064897679078
Gaussian Parametric Summary	0,0000015709059418

3.2. Інтерполяція зі зміною варіативного параметру α

Як можна побачити з попередніх досліджень, у випадку змінного кроку метод Гауса дає більш точні результати порівняно з класичним поліномом Лагранжа. Але і ці результати можна покращити, тобто зменшити похибку обчислень. Для цього потрібно проаналізувати вплив варіативного параметра на результати інтерполяції. Таким варіативним параметром є параметр α.

Розглянемо інтерполяційну формулу Гауса:

$$G(x) = \tilde{y}_1 e^{-\alpha(x-x_1)^2} + \tilde{y}_2 e^{-\alpha(x-x_2)^2} + \ldots + \tilde{y}_n e^{-\alpha(x-x_n)^2},$$

де α — варіативний коефіцієнт. У класичному випадку цей коефіцієнт записують:

$$\alpha = \frac{\pi(n-1)}{(x_{\max} - x_{\min})^2},$$

де x_{max}, x_{min} — максимальне та мінімальне значення аргумента x.

У попередніх дослідженнях [27] був використаний саме такий вигляд коефіцієнту α. В даній роботі було експериментально перевірено та досліджено можливість впливати на похибку, змінюючи коефіцієнт α.

Для зменшення похибки обчислень було запропоновано вводити цей коефіцієнт довільно і стежити за зміною похибки. Отож, було отримано оптимальні значення варіативного коефіцієнту для різних елементарних алгебричних функцій.

Крок для таких досліджень було вирішено обрати тим самим чином, як описано у підрозділі 3.1., адже реальні дані зазвичай не подаються з фіксованим кроком.

На рисунках 3.15-3.17 наведено результати інтерполяції елементарних алгебраїчних функцій з такими значеннями параметру α, за яких результат має найменшу похибку.

$$y = \frac{1}{x}$$

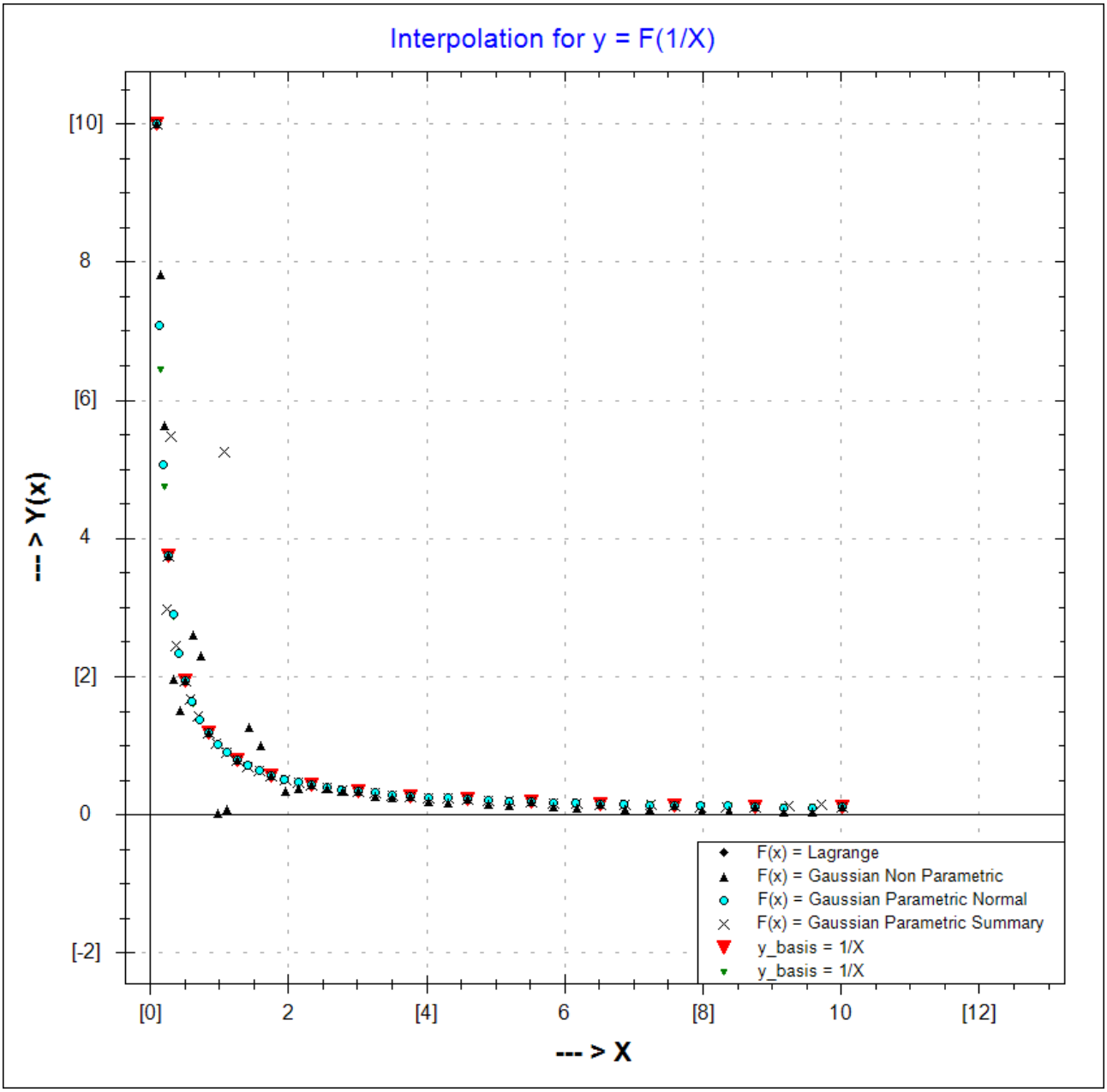

Рис. 3.15. — Інтерполяція функції $y = \frac{1}{x}$ зі змінним кроком та α

$X \in [0,1;10,1]$. Кількість точок = 15
Крок = [1 = 0,17 2 = 0,25 3 = 0,33 4 = 0,42 5 = 0,50 6 = 0,58 7 = 0,67 8 = 0,75 9 = 0,83 10 = 0,92 11 = 1,00 12 = 1,08 13 = 1,17 14 = 1,25 15 = 1,33]

Оцінка похибки алгоритмів:

Lagrange	23097,1105536787000000
Gaussian Non Parametric	0,0029433128697788
Gaussian Parametric Normal	0,0001845882267589
Gaussian Parametric Summary	0,0004442654023632

$$y = \sqrt{x}$$

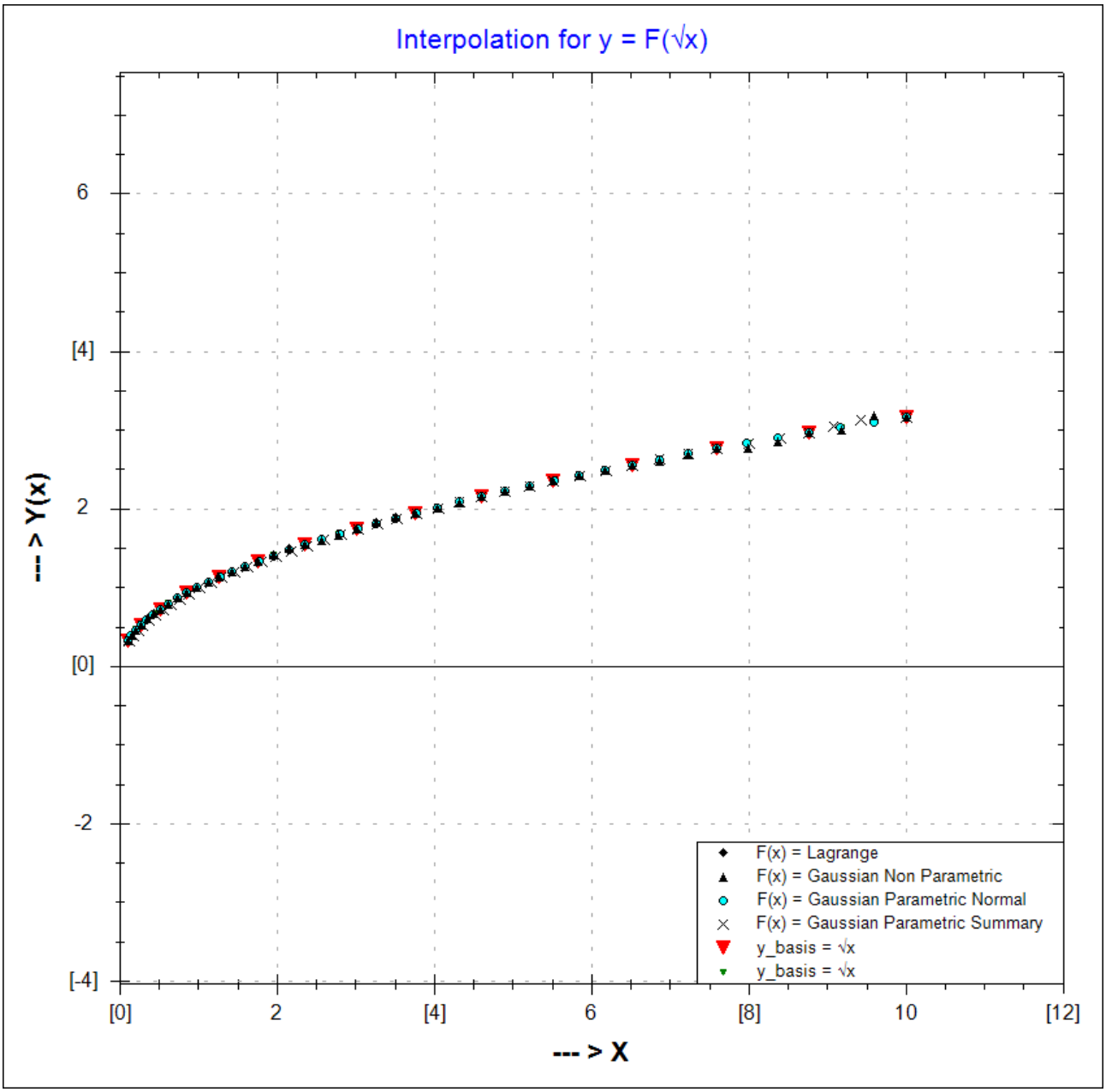

Рис. 3.16. — Інтерполяція функції $y = \sqrt{x}$ зі змінним кроком та α

X є [0,1;10,1]. Кількість точок = 15
Крок = [1 = 0,17 2 = 0,25 3 = 0,33 4 = 0,42 5 = 0,50 6 = 0,58 7 = 0,67 8 = 0,75 9 = 0,83 10 = 0,92 11 = 1,00 12 = 1,08 13 = 1,17 14 = 1,25 15 = 1,33]

Оцінка похибки алгоритмів:

Lagrange	5,8568776518629900
Gaussian Non Parametric	0,0008124876223612
Gaussian Parametric Normal	0,0000061058093832
Gaussian Parametric Summary	0,0001628668864064

$$y = \lg(x)$$

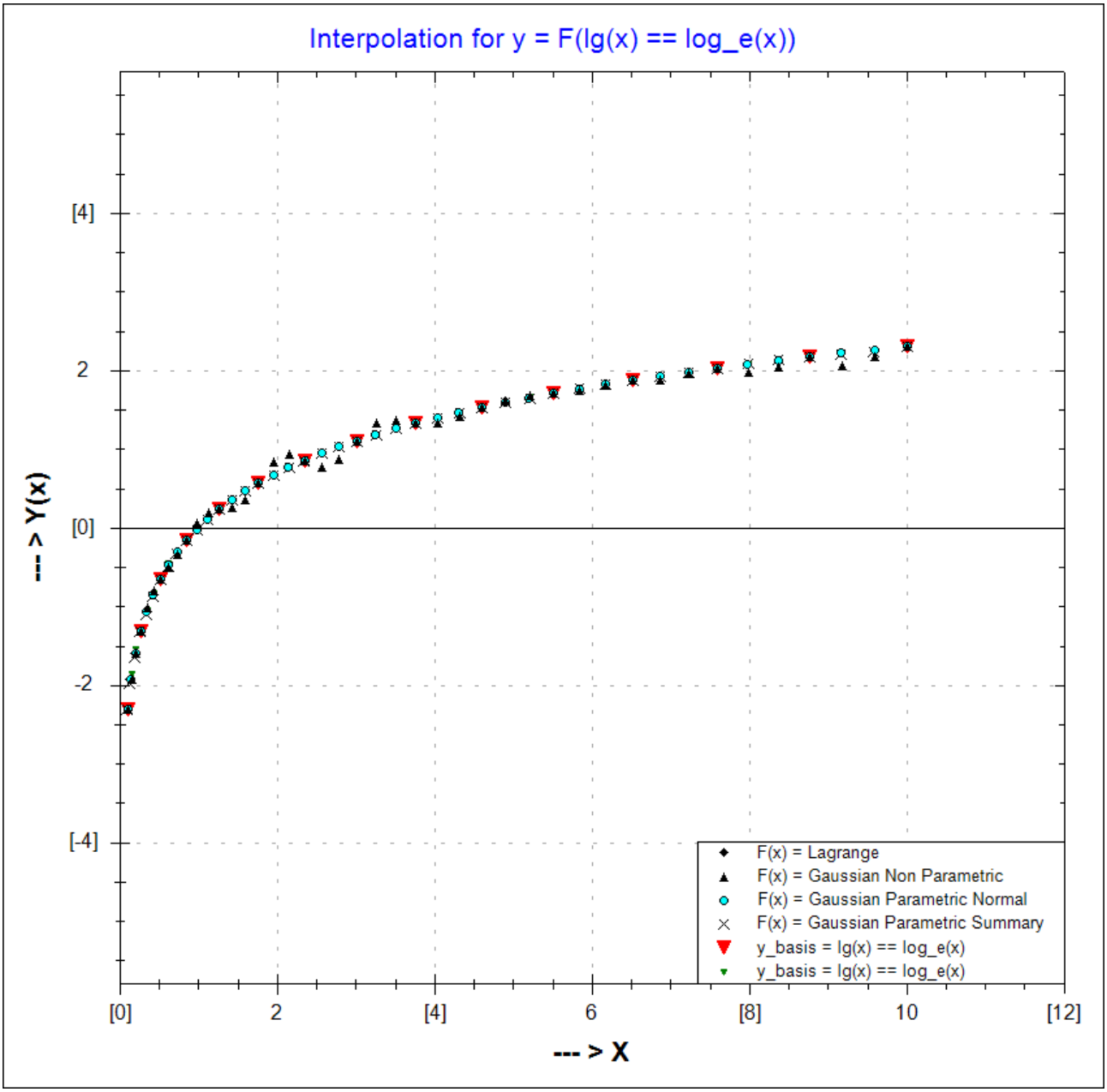

Рис. 3.17. — Інтерполяція функції $y = \lg(x)$ зі змінним кроком та α

$X \in [0,1;10,1]$. Кількість точок = 15
Крок = [1 = 0,17 2 = 0,25 3 = 0,33 4 = 0,42 5 = 0,50 6 = 0,58 7 = 0,67 8 = 0,75 9 = 0,83 10 = 0,92 11 = 1,00 12 = 1,08 13 = 1,17 14 = 1,25 15 = 1,33]

Оцінка похибки алгоритмів:

Lagrange	1788,0239165818600000
Gaussian Non Parametric	0,0017184070212567
Gaussian Parametric Normal	0,0000565851388933
Gaussian Parametric Summary	0,0001835633920919

3.3. Інтерполяція експериментальних даних

Як відомо, методи інтерполяції використовують, як правило, для отримання проміжних значень функцій, які задані таблицею. Для перевірки роботи алгоритму інтерполяційної функції Гауса на експериментальних даних було запропоновано провести інтерполяцію на прикладі розповсюдження захворювання COVID-19 на території України та Італії.

Дані для обробки були отримані з інтернет-сайту https://pomber.github.io/covid19 [4] , шляхом звернення до сайту, використовуючи *HTTP GET* запит. Даний ресурс збирає та оновлює дані двічі на день з таких джерел, як:

- *World Health Organization (WHO);*
- *DXY.cn. Pneumonia. 2020;*
- *BNO News;*
- *National Health Commission of the People's Republic of China (NHC);*
- *China CDC (CCDC);*
- *Hong Kong Department of Health;*
- *Macau Government;*
- *Taiwan CDC.*

За вузли інтерполяції були обрані дані через кожні два дні. Для обрахунку похибки використовувались значення, які не увійшли в задані вузли. Після отримання результатів система дозволяє змінювати похибку за рахунок використання довільного варіативного коефіцієнта α.

Приклади роботи алгоритму інтерполяційної функції Гауса на прикладі розповсюдження захворювання COVID-19 за період з 22.03.2020 по 07.05.2020 на території України та Італії представлено на рисунках 3.18-3.21.

Графіки представлені двох типів:

А) кількість всіх випадків захворювання за весь час;

В) кількість нових захворювань за день.

Статистику для України можна побачити на рисунках 3.18-3.19.

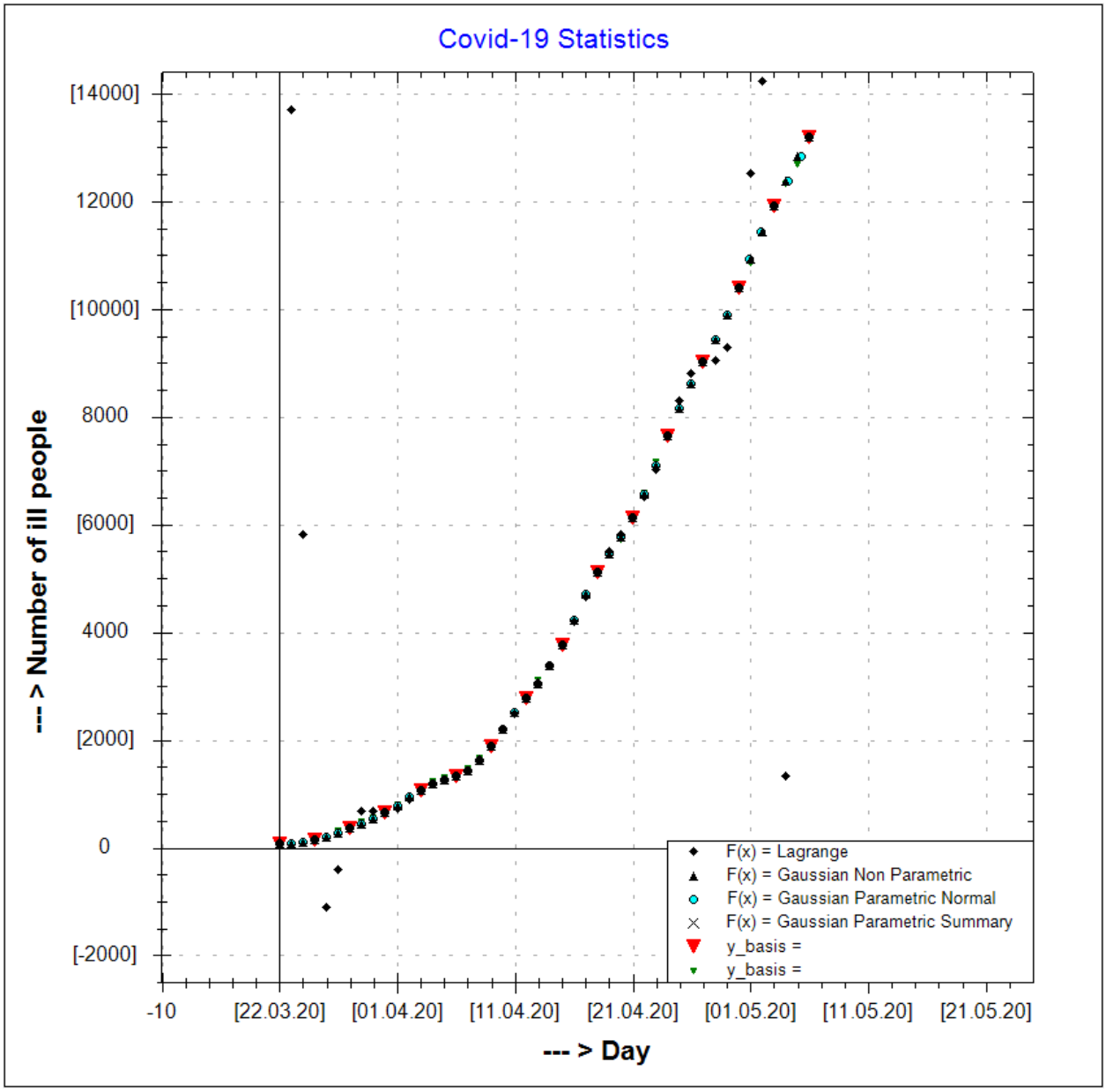

Рис. 3.18. — Інтерполяція функції COVID-19 на території України за період з 22.03.2020 по 07.05.2020. Тип графіку — A.

X є [0,1;10,1]. Кількість точок = 15

Оцінка похибки алгоритмів:

Lagrange	0,5891638014699470
Gaussian Non Parametric	0,0000103527652694
Gaussian Parametric Normal	0,0000103527652694
Gaussian Parametric Summary	0,0004251843459873

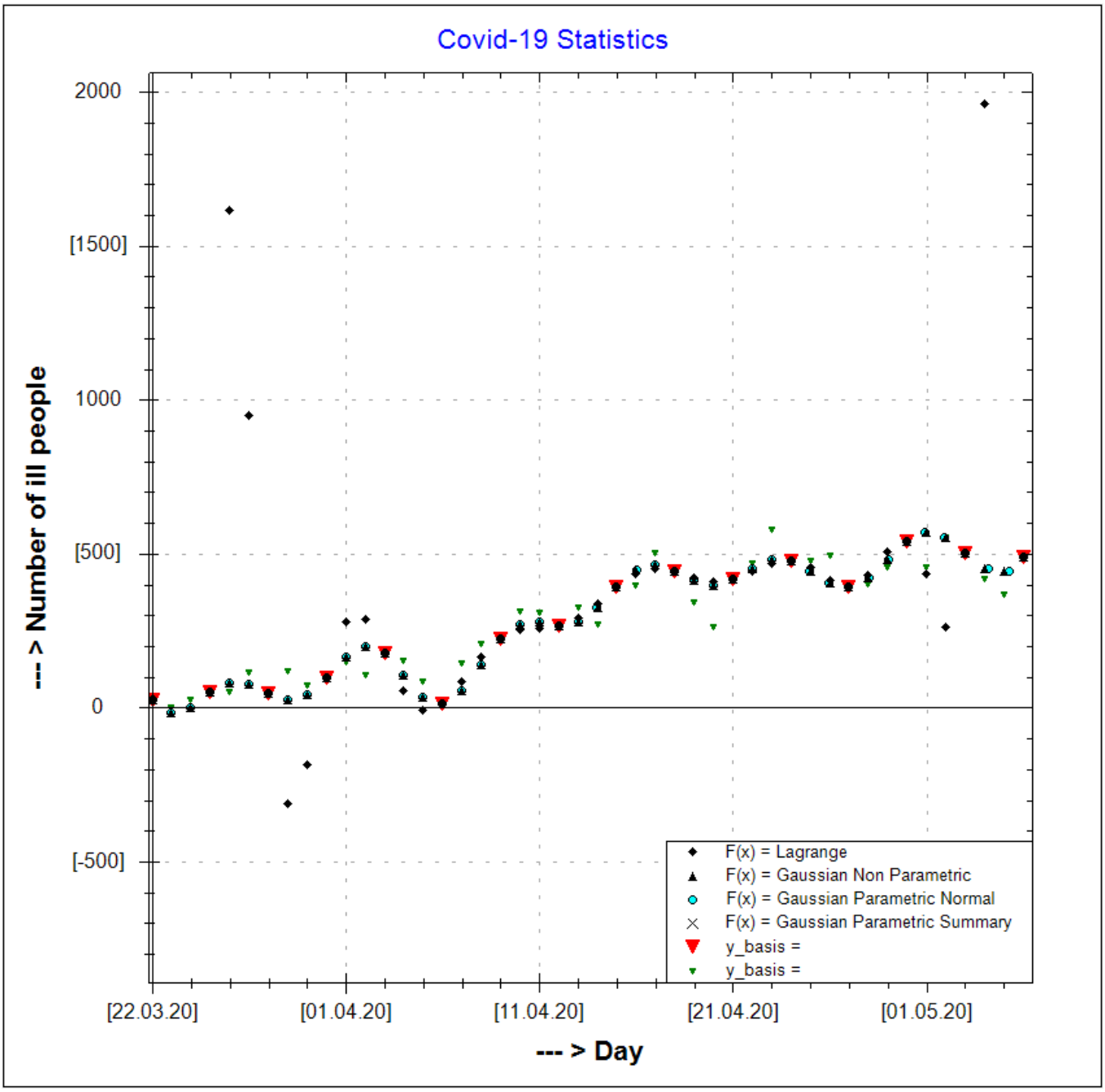

Рис. 3.19. — Інтерполяція функції COVID-19 на території України за період з 22.03.2020 по 07.05.2020. Тип графіку — B.

X є [0,1;10,1]. Кількість точок = 15

Оцінка похибки алгоритмів:

Lagrange	23,803934394300500
Gaussian Non Parametric	0,0113573096582980
Gaussian Parametric Normal	0,0113573096582505
Gaussian Parametric Summary	0,0239221080991645

Статистику для Італії можна побачити на рисунках 3.20-3.21.

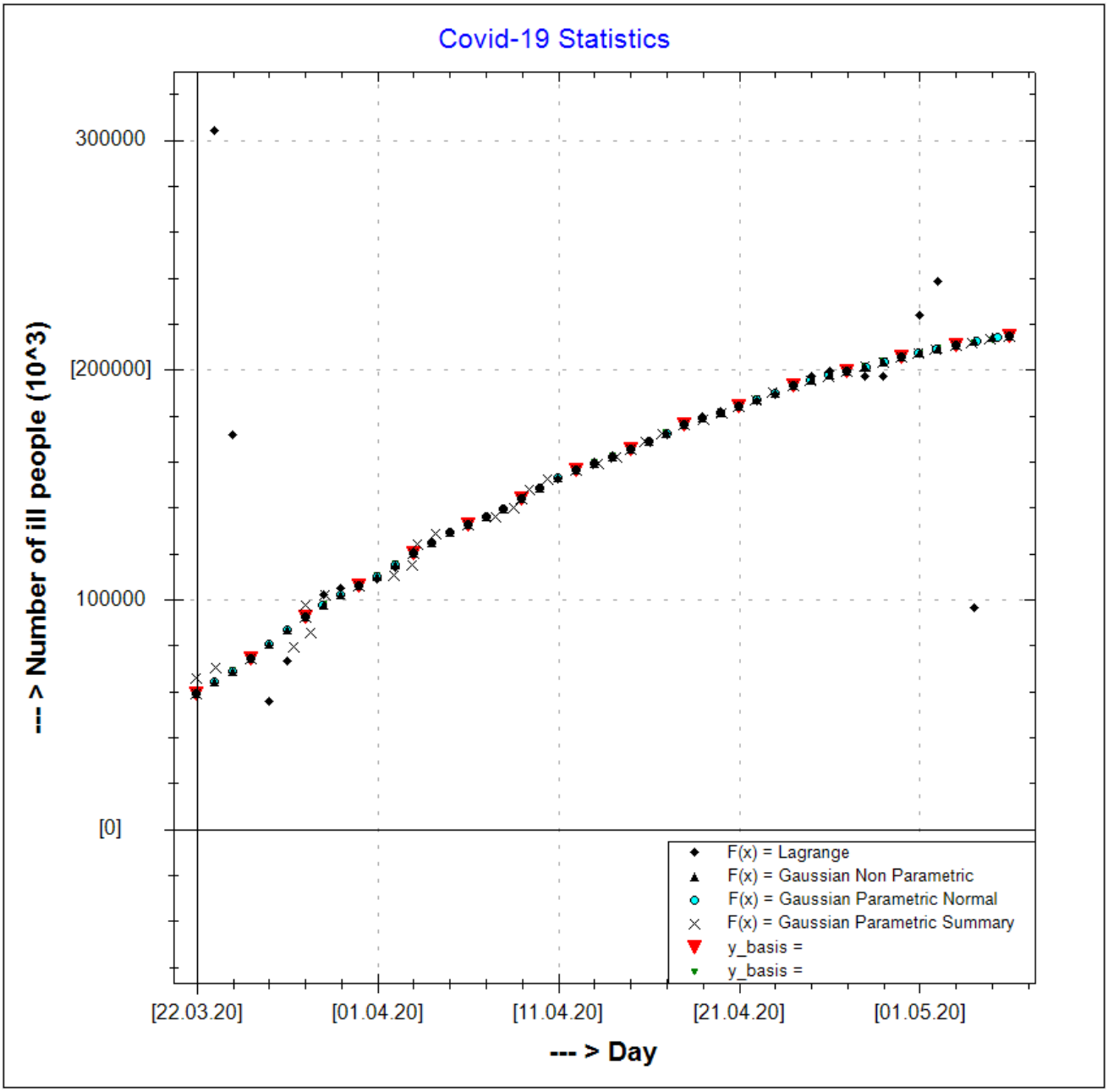

Рис. 3.20. — Інтерполяція функції COVID-19 на території Італії за період з 22.03.2020 по 07.05.2020. Тип графіку — A.

X є [0,1;10,1]. Кількість точок = 15

Оцінка похибки алгоритмів:

Lagrange	0,1124218547921890
Gaussian Non Parametric	0,0000023435417150
Gaussian Parametric Normal	0,0000023435417150

Gaussian Parametric Summary	0,0000068336833620

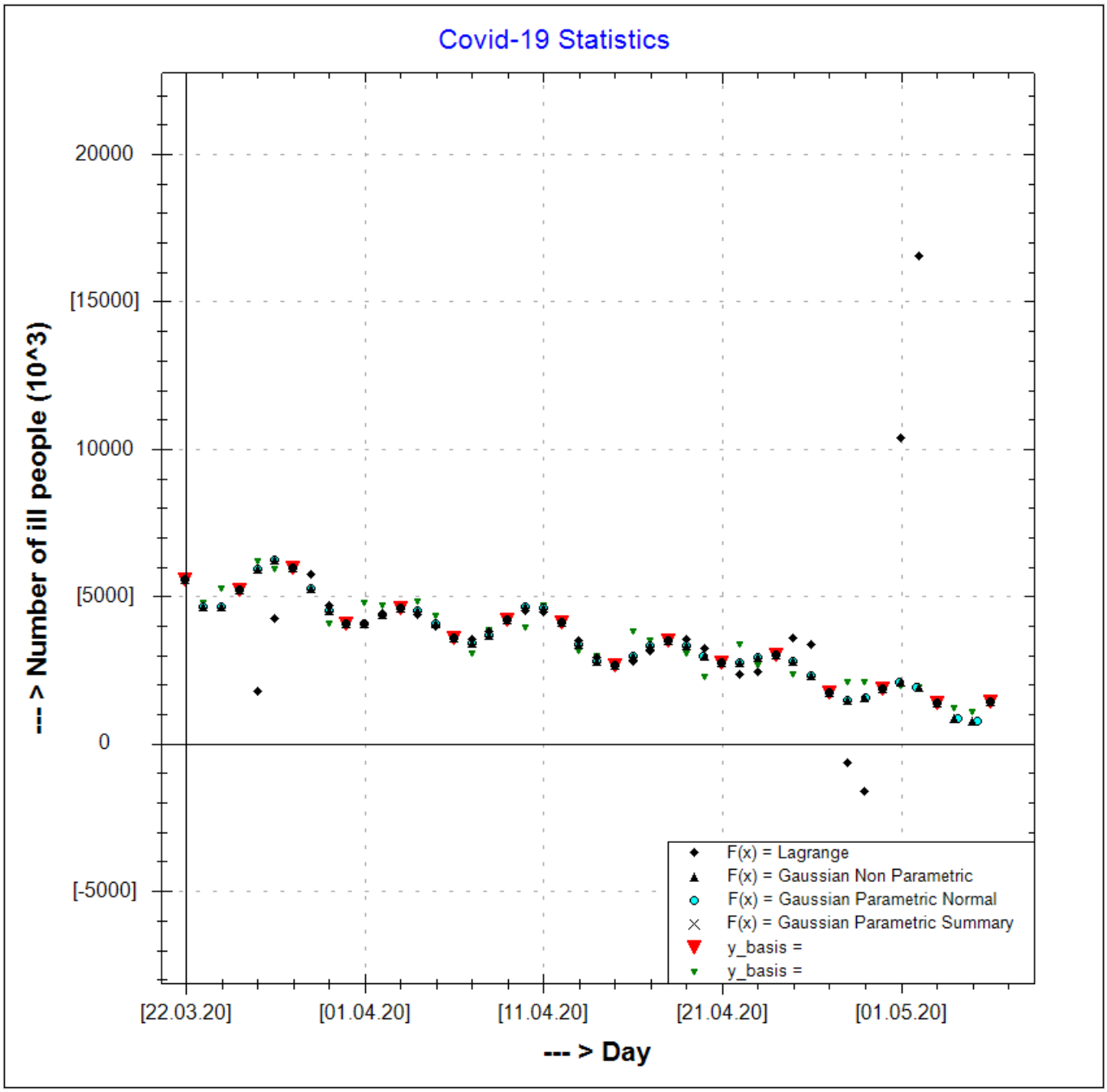

Рис. 3.21. — Інтерполяція функції COVID-19 на території Італії за період з 22.03.2020 по 07.05.2020. Тип графіку — B.

X є [0,1;10,1]. Кількість точок = 15

Оцінка похибки алгоритмів:

Lagrange	20,672853372426500
Gaussian Non Parametric	0,0045231765060441
Gaussian Parametric Normal	0,0045231765060407

Gaussian Parametric Summary 0,0125488148332005

За результатами комп’ютерного експерименту можна зробити такі висновки.

За умови великої кількості вузлів інтерполяції (більше 15) і змінного кроку, інтерполяції класичні методи інтерполяції мають великі відхилення від справжнього результату, а тому необхідно застосовувати інші методи, наприклад, інтерполяційну функцію Гауса.

Похибку методу Гауса можна зменшувати за допомогою впливу на варіативний коефіцієнт α.

Метод Гауса добре себе проявив при роботі з масивами експериментальних даних, що було підтверджено на прикладі інтерполяції розповсюдження захворювання COVID-19 на території України та Італії.

4. ОБҐРУНТУВАННЯ ЗАСОБІВ РЕАЛІЗАЦІЇ СИСТЕМИ

Для реалізації програмного продукту було обрано програмне середовище *Microsoft Visual Studio 2020,* фреймворк розробки *Microsoft .NET Framework* та мову програмування *C#.*

4.1. Обґрунтування вибору середовища для розробки

Існує декілька придатних до використання середовищ розробки програмного забезпечення на мові програмування *C#* з використанням *Microsoft .NET Framework* [23]. Такі середовища є: *Microsoft Visual Studio, Project Rider, Eclipse, Visual Studio Code* та *MonoDevelop.*

Було обрано середовище *Microsoft Visual Studio* тому, що:

- офіційне середовище від компанії *Microsoft*;
- є безкоштовна версія *"Community edition"*, функціонал якої повність покриває вимоги проекту, які описані у розділі 1;
- підтримує платформи *.NET*. *Visual Studio* має широкі можливості по розробці додатків під Windows, в тому числі в .NET-сегменті [24].

До мінусів інших вказаних середовищ розробки можна віднести:

- частина функціоналу знаходиться в стадії розробки. Погано протестовано, маленьке ком'юніті;
- вартість. Для розробки навіть тестового програмного забезпечення необхідно заплатити не менш як 130$.
- низька функціональність у відношенні до *Visual Studio.* Не дивлячись на достатньо велику підтримку, деякі з вище вказаних продуктів не підходять для складних проектів.

4.2. Обґрунтування вибору платформи *Microsoft .NET Framework*

Вибір середовища програмування — це досить важлива дилема ще перед розробкою програмного забезпечення. Платформа для розробки даного програмного забезпечення було вирішено обрати *Microsoft .NET Framework* через низку переваг, а саме:

- простота у використанні [16];
- платформа має весь необхідний набір методів та модулів для розробки даного програмного забезпечення з вимогами, вказаними у параграфі 1;
- швидкість розробки. Платформа добре структурована і має коректну документацію по всім модулям та методам. Більшість сторінок документації переведені на українську мову;
- підтримка багатьох мов програмування, серед які найбільш відомі [5]: *C#, VB.NET, C++, F#* — та інші;
- кросплатформеність: .NET платформа, що може бути перенесена. (Проте *.NET Framework* працює лише на операційній системі *Windows*.) Наприклад, *Windows, MacOS* та *Linux* підтримують *.NET Core*. Крім того, на цій платформі можна розробляти програми для: *Windows, MacOS, Linux, Android, IOS* та інші;
- бібліотека класів *.NET* єдина для усіх мов, які вона підтримує;
- одна з найкращих підтримок динамічних *WEB*-сторінок та баз даних [10];
- платформа *.NET* поставляється разом із *Microsoft Visual Studio*;

4.3. Обґрунтування вибору мови програмування *C#*

Від вибору мови програмування залежить досить багато, тому обирати потрібно почати з аналізу найбільш популярних на даний момент мов: *C++*, *C#* та *Java*.

C++ було розроблено на основі мови програмування C ще на початку 80-х років В.Страуструпом, і це був вдалий експеримент, який набув популярності:

структурована, об’єктно-орієнтована, компілюєма та має потенціал для розвитку — C++ продовжує свій розвиток навіть зараз, наприклад, специфікації C++17 було опубліковано у грудні 2017, і одразу була розпочата робота над стандартом C++2020. У C++ досить багато переваг:

— C++ економічна мова програмування. Згідно з дослідженням [1], C++ є другою за споживанням пам’яті;
— C++ енергоощадлива мова програмування. Згідно з дослідженням [1], C++ є третьою за енергозбереженням;
— C++ універсальна мова програмування [28]. Компілятори C++ наявні для кожної операційної системи, більшість програм легко переносяться з платформи на платформу [22], для цієї мови програмування розроблено багато бібліотек. Це обумовлено тим, що C++ використовується для мікроконтролерів [21], роботів, мобільних додатків [17], ігор [31], систем моделювання [18], прогнозування, обробки статистики і в нейронних мережах.

Згідно з думкою Г.Шилдта [32]: «Мова C ++ - єдина (з найзначніших) мова програмування, яку може освоїти будь-який програміст».

Проте є у C++ недоліки, через які ця мова не була обрана для реалізації даного програмного забезпечення:

— довга компіляція для великої кількості коду: погана підтримка модульності, тобто підключення інтерфейсу зовнішнього модуля робиться через вставку заголовного файлу *#include*, і при великій кількості заголовних файлів компіляція може сильно сповільнюватися, так як результуючий файл має великий обсяг;
— великий об’єм згенерованого машинного коду;
— мова C++ розроблялася сумісною із Сі, щоб можна було легко «перемикатися» з однієї мови на іншу, проте разом із цим C++ успадкував багато проблем Сі, серед яких складний синтаксис, складність портування програми при роботі з іншим компілятором (код може погано працювати на іншій платформі або не працювати зовсім).

Крім C++, була розглянута мова програмування *Java*. Мова *Java* була розроблена компанією *Sun Microsystems Inc* ще у далекому 1995 році (у подальшому була придбана компанією *Oracle*, яка є власником і зараз) [33]. Java задумувалася платформо-незалежною мовою програмування (на той час це був недосяжний ідеал) та схожою на мову C++ (наприклад, був успадкований принцип ООП. У Java наявні такі плюси:

— кросплатформена сумісність;
— об'єктна орієнтованість (всі дані і дії групуються в класи об'єктів);
— автоматичне керування пам'яттю.

Проте, не дивлячись на усі плюси, у *Java* є значні недоліки, через які ця мова не була обрана:

— довга компіляція [30];
— відсутня підтримка низькорівневого програмування (наприклад, відсутні вказівники);
— потребує багато пам'яті.

Також була розглянута та обрана мова програмування *C# (C Sharp)*. Мова *C#* досить молода (вона була створена компанією *Microsoft* у 2000 році). У той час була популярна *Java*, і *Microsoft* вирішили створити свою мову, схожу на *Java*, яка також призначалася для веб-розробки і має схожі синтаксичні властивості [26]. Ще частково Сі Шарп запозичив у C++ та *Visual Basic*. Крім цього, ця мова була створена спеціально для роботи з фреймворком *.NET*. Переваги, через які була обрана саме ця мова програмування, наступні:

— вказівники та повний доступ до пам'яті при необхідності;
— підтримка *Microsoft*, яка є одною із найбільших IT-компаній світу, а C# — це її мова програмування;
— власна середа розробки (*.NET Framework*);
— наявність event та, відповідно, вбудованої обробки подій;
— *JIT*-компіляція. Увесь код транслюється на проміжну мову *IL*, яка, у свою чергу, перетворюється на машинний код під час виконання

програми *JIT* зразу на комп’ютері. Так як кінцева компіляція проводиться на конкретному комп’ютері, то вихідна програма адаптована саме під нього, що збільшує продуктивність;

— багато синтаксичного цукру (синтаксичний цукор – це конструкції, які створені для полегшення написання та розуміння програмного коду);
— підтримка багатопоточності;
— об’єктно-орієнтованість;
— кросплатформеність.

Проте не обійшлося без недоліків:

— майже відсутній захист коду. Якщо програма на C++ може бути декомпільована лише при глибоких знаннях асемблеру, то початковий код програми на C# можна досить легко отримати;
— саме через *JIT*-компіляцію програма компілюється у машинний код прямо під час роботи програми, тобто може бути гальмування;
— чутливість до реєстру.

Згідно з проведеними дослідженнями, для реалізації програмного продукту було обрано мову програмування C# на платформі *.NET Framework.*

5. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

У цьому розділі буде описано архітектуру та основні модулі програмного продукту.

5.1. Архітектура додатку

Побудова архітектури є одним з найважливіших етапів розробки програмного додатку: це представлення програмного забезпечення на першому етапі програмування, яке дає змогу оцінити цілу картину майбутнього додатку зі всіма взаємозв'язками між його компонентами. Побудова архітектури також необхідна для того, щоб пришвидшити розробку додатку та виправити основні недоліки майбутньої реалізації до початку програмування. При розробці додатку для даної роботи було створено наступну архітектуру (рис. 5.1-5.4).

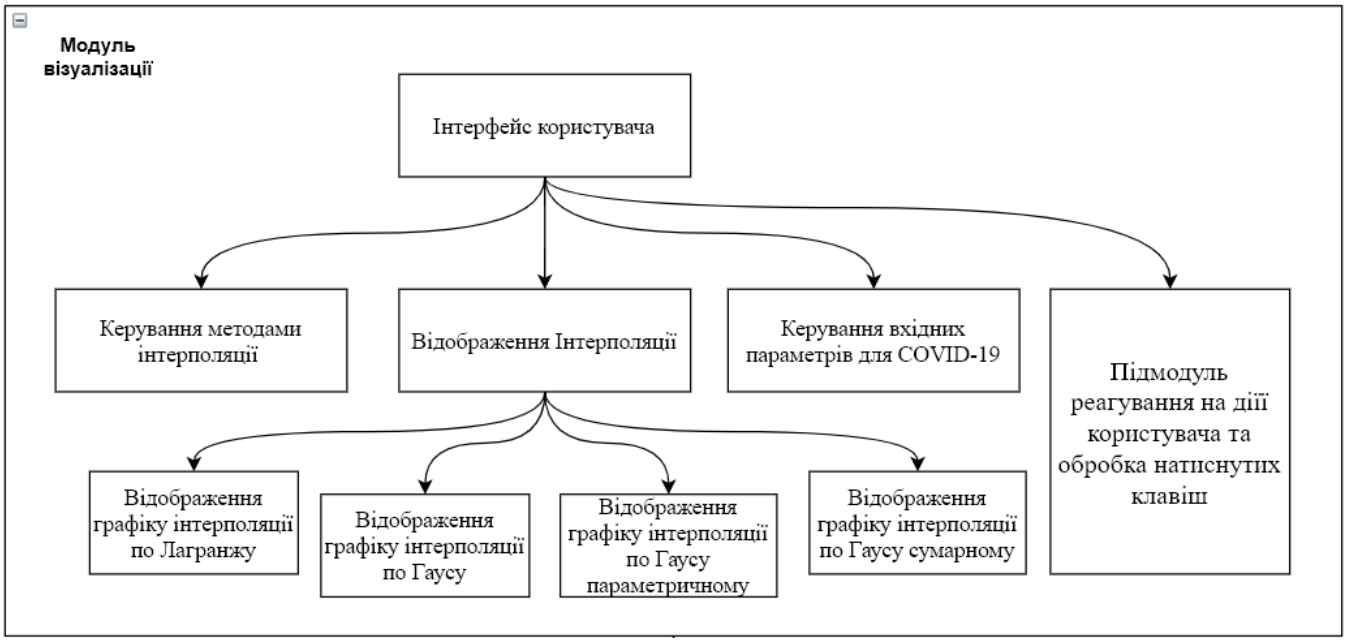

Рис. 5.1. — Архітектура додатку. Частина 1

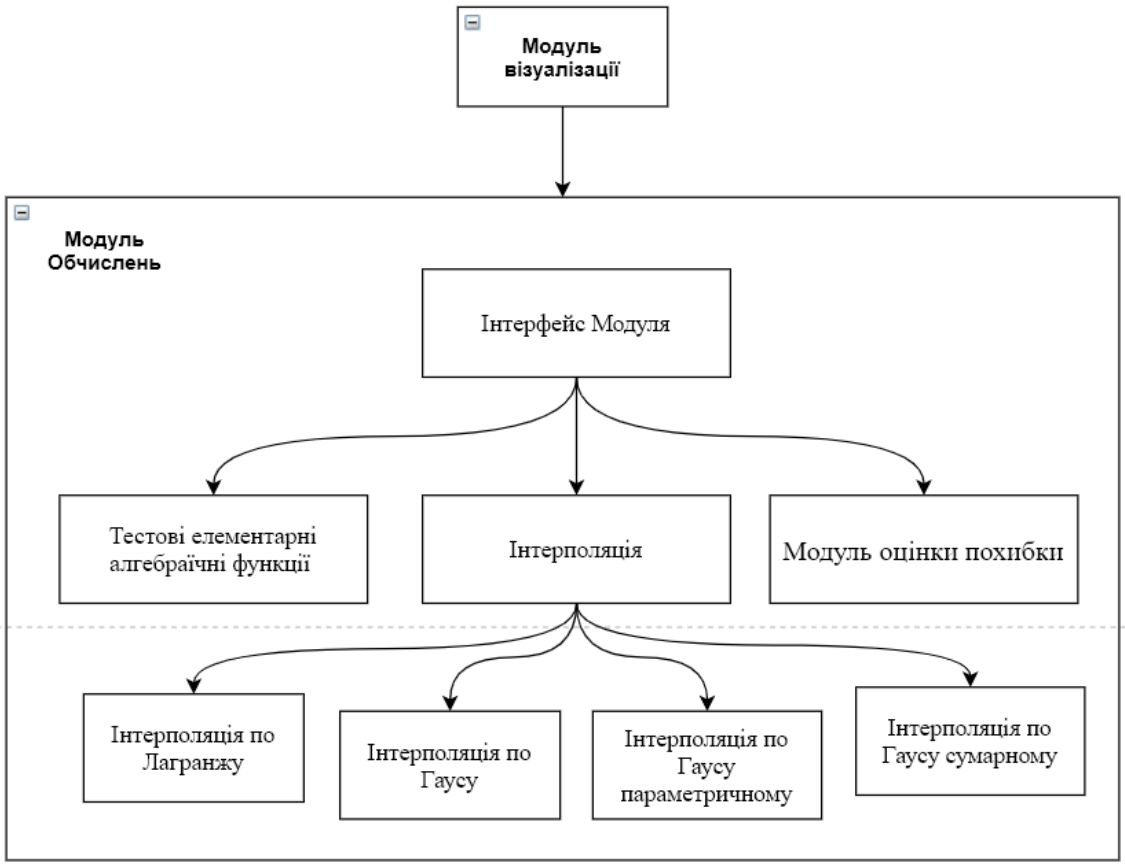

Рис. 5.2. — Архітектура додатку. Частина 2

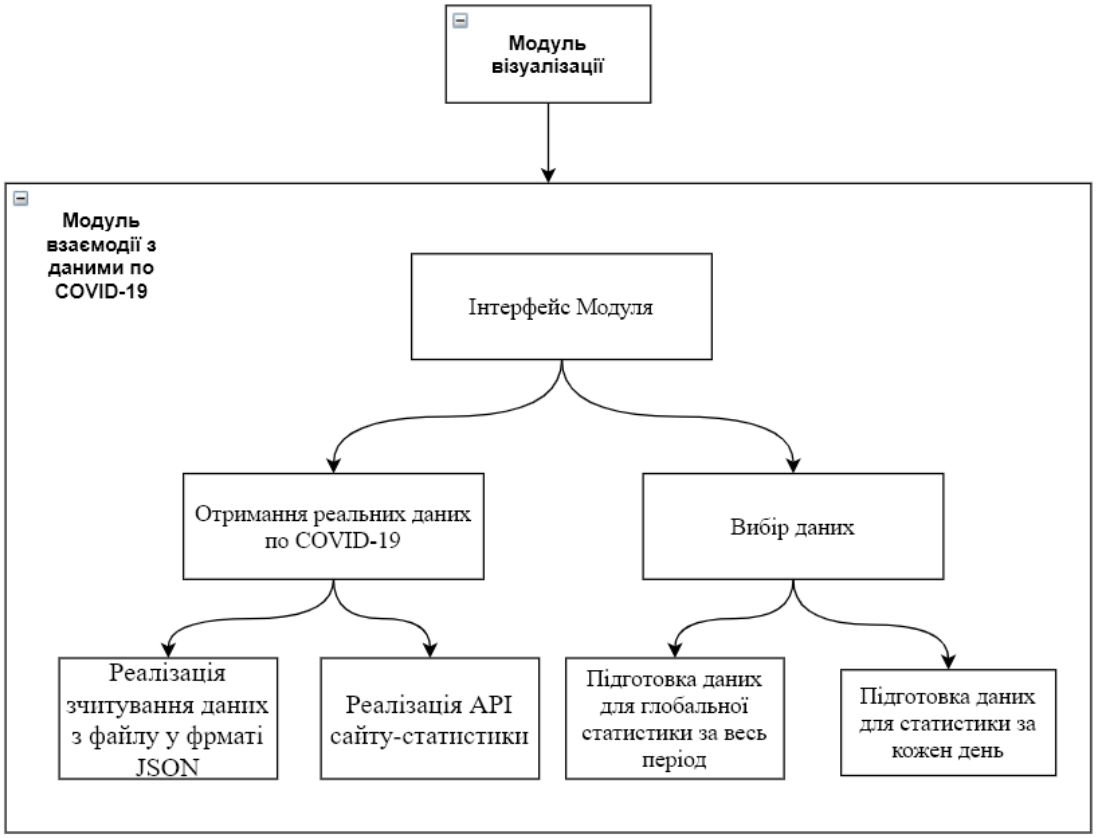

Рис. 5.3. — Архітектура додатку. Частина 3

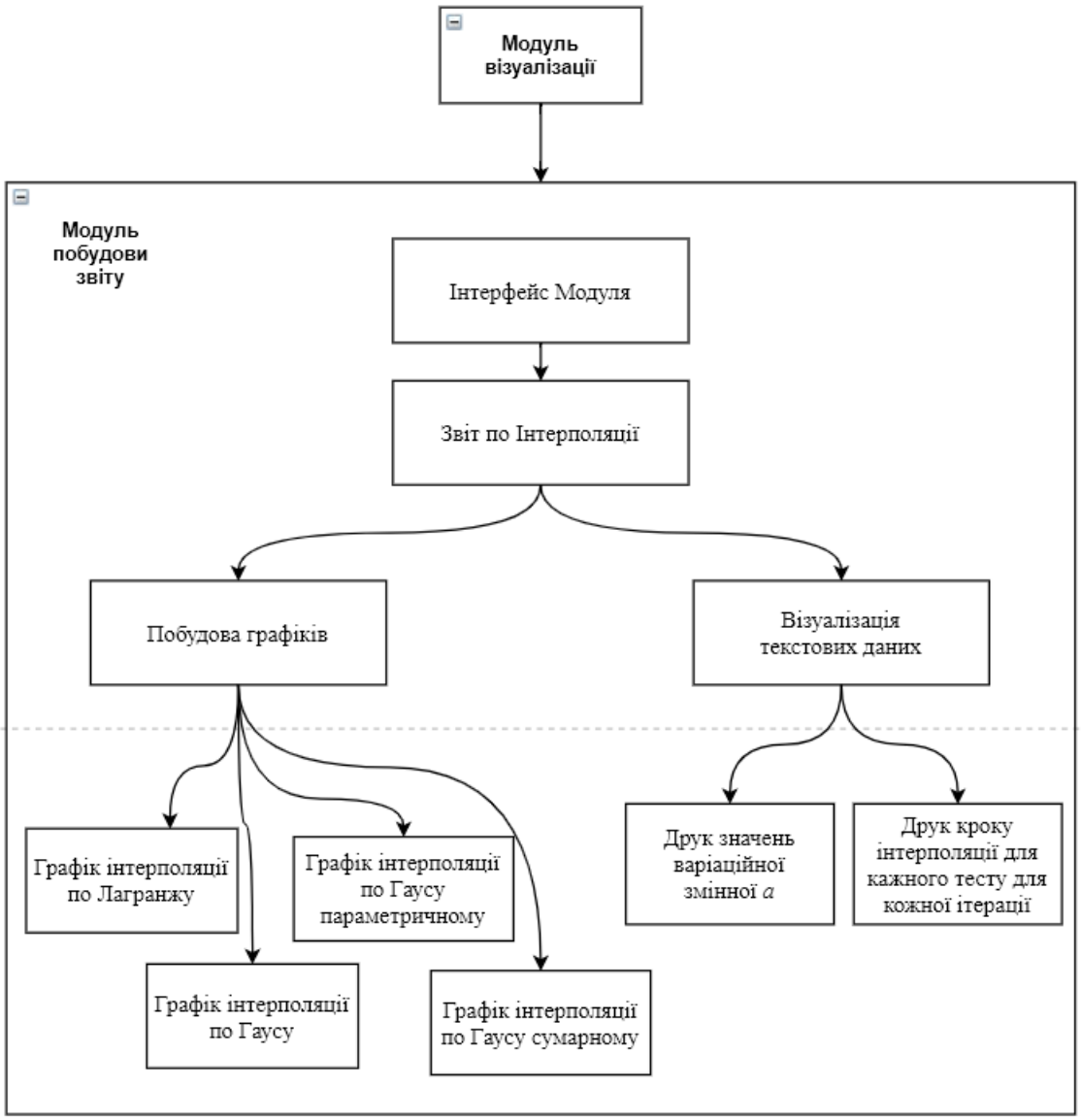

Рис. 5.4. — Архітектура додатку. Частина 4

Система складається з чотирьох основних модулів. Такими є:

— модуль візуалізації (рис. 5.1);

— обчислювальний модуль (рис. 5.2);

— модуль взаємодії з даними по захворюванню COVID-19 (рис. 5.3);

— модуль побудови звіту (рис. 5.4).

Кожен модуль системи є окремим та самостійним. Його використання відбувається через відповідний інтерфейс.

Модуль візуалізації виконує роль відображення даних, а також роль введення даних таких як: тип кроку, значення варіаційної змінної для інтерполяції методами

Гаус-функцій, вибір функцій для проведення замірів, вибір країн та вибір періоду, за який треба зібрати дані по захворюванню COVID-19.

Обчислювальний модуль відповідає за методи інтерполяції, побудову базисних та тестових наборів даних, оцінки похибки результатів.

Модуль взаємодії з даними по захворюванню COVID-19 реалізує взаємодію з сайтом зі статистикою, виокремлює потрібні дані, фільтрує дані та виконує приведення даних до потрібного виду, а саме: побудова каркасу точок з кількістю всіх випадків захворювання за весь час та кількість нових захворювань за день.

Модуль побудови звіту обробляє проаналізовані результати інтерполяції методами, наведеними вище, та будує звіт у форматі *Microsoft Word Document (DOCX).* Також, модуль друкує інформацію про значення варіативних змінних, які були використані та оцінку алгоритму для кожної протестованої функції.

5.2. Опис підпроцесів додатку

На сьогодні майже неможливо уявити собі прикладний програмний додаток, який використовує лише один процес.

У комп’ютерній архітектурі багатопотоковість або багатопроцесовість — це можливість центрального процесора *(CPU)* або одного ядра у багатоядерному процесорі одночасно виконувати декілька процесів або потоків.

При розробці даного додатку виникла необхідність використовувати підпроцеси тому, що одночасно повинно виконуватись декілька речей: взаємодія програми з користувачем та обробка даних. Обробкою даних у даному додатку будемо називати побудову звітів, обчислення точок інтерполяції, викачування даних з мережі, перетворення даних.

Таким чином, було реалізовано три фонові підпроцеси (рис. 5.5):

— головний процес;

— обчислювальний процес;

— процес побудови звітів.

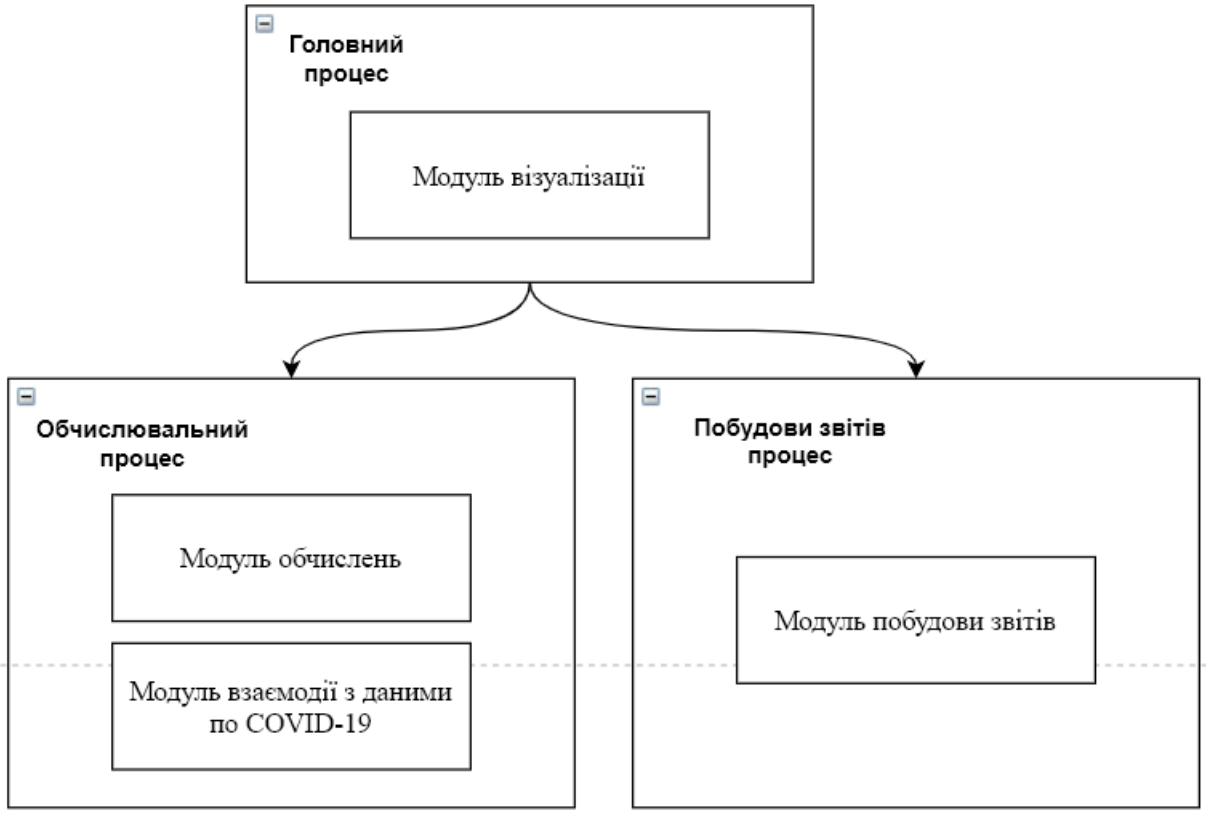

Рис. 5.5. — Архітектура підпроцесів

Головний процес обробляє модуль візуалізації даних та введення/виведення даних від користувача. Обчислювальний процес обробляє модуль обчислення та модуль взаємодії з даними по захворюванню COVID-19. Процес побудови звіті виконує модуль побудови звітів.

6. РОБОТА КОРИСТУВАЧА З ПРОГРАМОЮ

Для використання розробленого продукту необхідно мати комп'ютер з наступними мінімальними вимогами:

— процесор *Intel ® Core ™ 2 / 2 Quad / Pentium ® / Sempron ™ / Celeron ® / Xeon™ чи AMD 6 / Turion ™ / Athlon ™ / Duron ™* / з тактовою частотою вище за 3 Гц;

— операційна система сімейства *MS Windows*;

— об'єм оперативної пам'яті — 4 Гбайт, рекомендовано 16 Гбайт;

— жорсткий диск не менше ніж 1 Тбайт;

— монітор Acer ConceptD HDR G-SYNC Compatible 27 дюймів з частотою оновлення матриці не менше ніж 144 Гц ;

— комп'ютерна миш з підтримкою натискання двох типів клавіш: права кнопка миші, ліва кнопка миші та наявність колеса, для масштабування зображення. Рекомендовано мати додаткові клавіші для швидкої навігації по зображенню графіка;

— клавіатура може бути відсутня.

Інтерфейс програми показано на рисунках 6.1-6.5.

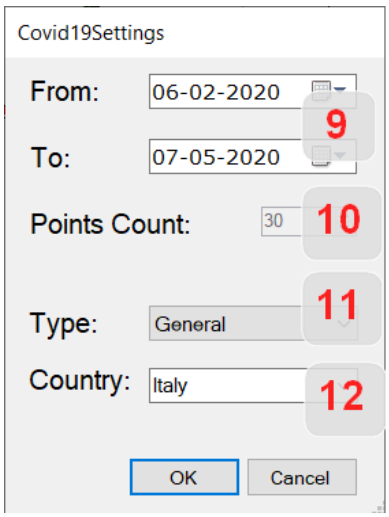

Рис. 6.1. — Інтерфейс користувача. Вікно налаштувань COVID-19

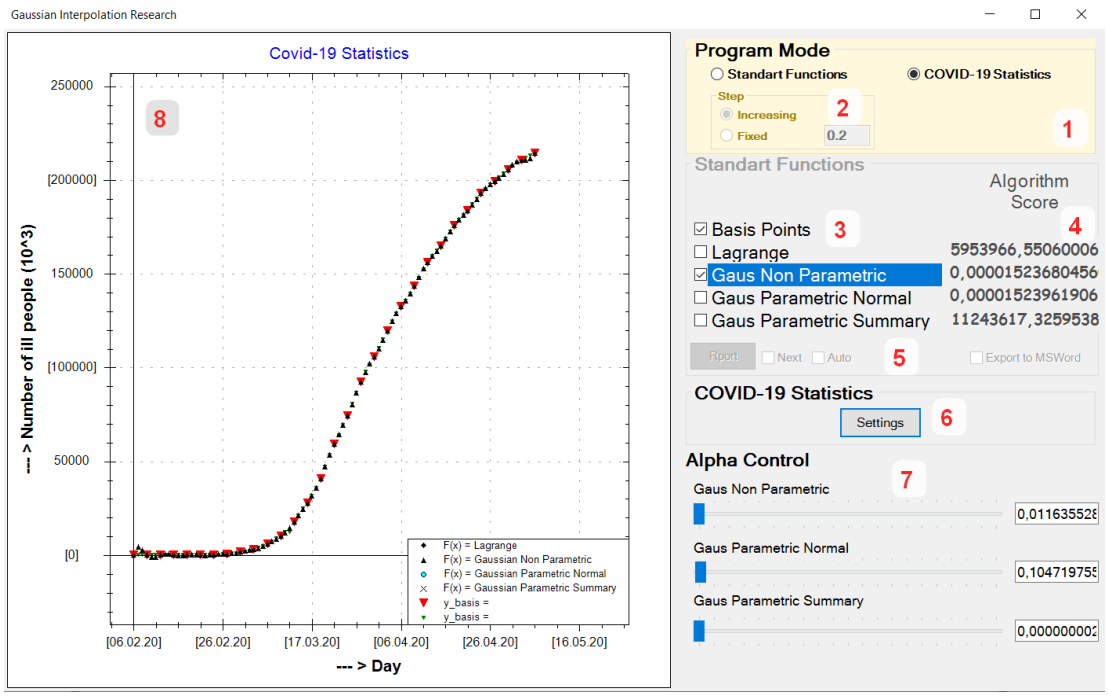

Рис. 6.2. — Інтерфейс користувача. Головне вікно

Червоним кольором позначені основні механізми для взаємодії з користувачем:

1) вибір типу даних: елементарні функції або статистика захворювання COVID-19;
2) вибір типу кроку: зростаючий або фіксований;
3) вибір типів інтерполяції, які будуть відображатись на графіку;
4) похибка кожного з методів інтерполяції для поточного графіку;
5) перемикач для переходу від однієї елементарної функції до іншої у режимі елементарних функцій, а також клавіша для автоматичного будування звіту в *MS WORD*;
6) клавіша для відкриття вікна налаштувань COVID-19;
7) повзунок для налаштування варіативних змінних α для методів Гауса;
8) область для перегляду результатів графіків, а також для керування графіками. Має наступний функціонал: збільшення та зменшення масштабу, переміщення, експорт у формати *JPEG* та *PDF*, друк, копіювання у буфер обміну, повернути масштаб за замовчуванням;
9) вибір періоду часу, за який брати дані для інтерполяції;

10) відображення кількості вузлів інтерполяції;

11) вибір типу статистики по захворюванню COVID-19;

12) вибір країни, для якої відбудеться інтерполяція.

Для того, щоб побудувати інтерполяцію по захворюванню COVID-19 потрібно натиснути на клавішу *OK* (рис. 6.3).

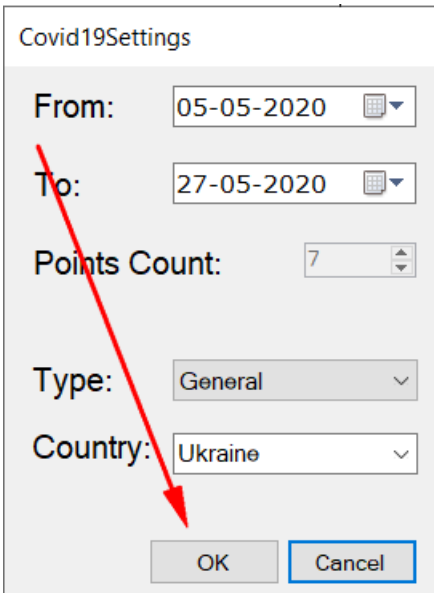

Рис. 6.3. — Інтерфейс користувача. Вікно налаштувань COVID-19

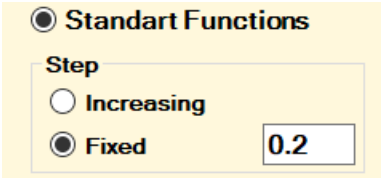

Рис. 6.4. — Вибір типу кроку інтерполяції

Для того, щоб обрати тип кроку інтерполяції: зростаючий або фіксований — користувачу необхідно натиснути лівою кнопкою миші на клавішу, вказану на рисунку (рис. 6.4). Також є можливість виставити величину фіксованого кроку.

На рисунку 6.5 представлена панель керування побудовою звіту.

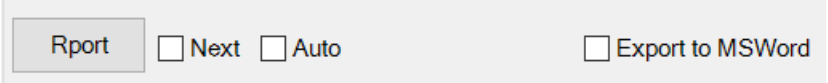

Рис. 6.5. — Панель керування побудовою звіту

Для того, щоб розпочати побудову звіту, необхідно натиснути клавішу *Report*. Для того, щоб звіт побудувався автоматично, необхідно обрати перемикач *Auto*. Для ручного керування та переходу до наступної тестової функції необхідно відтиснути перемикач *Auto* та натиснути перемикач *Next*. При натисненому перемикачу *Export to MS WORD* побудується у програмі *MS WORD* звіт і автоматично відкриється вікно зі звітом, після чого його можна зберегти.

ВИСНОВКИ

В результаті виконання бакалаврської роботи було розроблено програмний продукт на мові програмування C# для аналізу та демонстрації роботи різних алгоритмів інтерполяції шляхом побудови двовимірних графіків, підрахунку похибки кожного з алгоритмів з можливістю керування варіаційною змінною α для інтерполяції на основі Гаус-функцій.

Було проаналізовано результати інтерполяції на основі поліномів Ньютона, Лагранжа, кусково-лінійної інтерполяції, Гаус-функції, параметричної Гаус-функції, сумарної Гаус-функції. Описано переваги та недоліки даних методів, та надано рекомендації щодо використання.

За допомогою створеної системи було проведено аналіз чотирьох інтерполяційних методів: Лагранжа, кусково-лінійної інтерполяції, Гаус-функції, параметричної Гаус-функції, сумарної Гаус-функції на прикладі елементарних алгебраїчних функціях та на статистиці захворювання COVID-19 на території України та Італії. Також досягнуто зменшення похибки при інтерполяції на основі Гаус-функцій шляхом зміни варіативної змінної α. Наведено приклади зі стандартної змінною α та зі зміненою. Проаналізовано результати виконаного дослідження, проілюстровано переваги використання Гаус-методів інтерполяції на алгебраїчних функціях та реальних статистичних даних.

Результати роботи, а саме програмний засіб — вплив коефіцієнта згладжування на вигляд інтерполяційної функції Гауса, у вигляді запису на CD, а також документацію програмного супроводу, прийняті для використання в розробках спеціалізованого програмного забезпечення ТОВ «БІ-ХАБ».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Energy efficient programming languages [Електронний ресурс] // Programming. – 2009. – URL: https://jaxenter.com/energy-efficient-programming-languages-137264.html
2. Gorodetskiy M. Robotik integration in our lives / Gorodetskiy M. // Science and Technology of the XXI Century : the XIX All-Ukrainian Students R&D Conference Proceeding, (Kyiv, November 29, 2018) / National Technical University of Ukraine „Igor Sikorsky Kyiv Polytechnic Institute“. – Part III. – Kyiv, 2018. – p. 32-33
3. Gorodetskiy M. Robotik integration in our lives [Електронний ресурс] / 3. Gorodetskiy // Всеукраїнська студентська науково-практична конференція "Визначні досягнення в науці та техніці", (м. Київ, 15 листопада 2017 р). / Національний технічний університет України „Київський політехнічний інститут імені Ігоря Сікорського“ – 2017. – Режим доступу до ресурсу: http://www.kamts1.kpi.ua/node/2296.
4. Pombo R. JSON time-series of coronavirus cases (confirmed, deaths and recovered) per country - updated daily [Електронний ресурс] / Rodrigo Pombo // github.com. – 2020. – Режим доступу до ресурсу: https://pomber.github.io/covid19/timeseries.json.
5. Архипенков С. Хранилища данных. От концепции до внедрения — М.: Диалог-МИФИ, 2010. — 528 с.
6. Архіпкін О.П. Основні результати і направлення розвитку космічного моніторингу в Казахстані / О.П. Архіпкін, Г.Н. Сагатдинова. Сучасні проблеми дистанційного зондування Землі з космосу. — М.:Мир, 2013. — С. 292-302.
7. Бадаев Ю.И. Интерполяция на основе функций Гаусса [Текст] / Ю.И. Бадаев, Ю.В. Сидоренко // Сборник трудов III международной научно-

практической конференции "Современные проблемы геометрического моделирования": Тезисы докл. — Мелитополь, 5-7 июня 1996 г. — С.32-33.

8. Бадаєв Ю. І. Реалізація інтерполяційного методу Гаусс-функції та порівняльний аналіз / Ю. І. Бадаєв, Ю. В. Сидоренко. Прикладна геометрія та інженерна графіка. — К.:КДТУБА, 1998. — 27 с.

9. Бадаєв Ю.І. Реалізація інтерполяційного методу Гаус-функції та порівняльний аналіз [Текст] / Ю.И. Бадаев, Ю.В. Сидоренко // Прикладна геометрія та інженерна графіка — К.:КДТУБА, 1998, вип.63 — С.33-37.

10. Бирн Д. Microsoft SQL Server 6.5. Руководство администратора — М.: Лори, 2008. — 211 с.

11. Бронштейн И. Н. Справочник по математике для инженеров и учащихся вузов / И. Н. Бронштейн, К. А. Семендяев. — М.: Наука, 2007. — 708 с.

12. Городецький М.В. Варіанти інтерполяційної функції Гауса / М.В. Городецький, Ю. В. Сидоренко// Сучасні проблеми наукового забезпечення енергетики: Матеріали XVII Міжнародної науково-практичної конференції молодих вчених та студентів, м. Київ, 23–26 квітня 2019 р. У 2 т. –К. : "КПІ ім. Ігоря Сікорського", 2019. –Т.2. – С.87

13. Городецький М.В. Комп’ютерна стеганографія як технологічний засіб захисту конфіденційної інформації / М.В. Городецький, Л.І. Кублій // Молодий будівничий України №37. Збірник матеріалів XXII Міжнародної науково-практичної конференції молодих вчених, аспірантів та студентів “Молодіжна політика в контексті євроатлантичного вибору України”, 19 квітня 2017 р. — К.: КиМУ, 2017. — Т. 2. — С. 429-433.

14. Городецький М.В. Комп’ютерний стеганографічний захист інформації / М.В. Городецький, Л.І. Кублій // Сучасні проблеми наукового забезпечення енергетики: Матеріали XV Міжнародної науково-практичної конференції

аспірантів, магістрантів і студентів, м. Київ, 25-28 квітня 2017 р. — К.: НТУУ “КПІ ім. Ігоря Сікорського”, 2017. — Т. 2. — С. 110.

15. Городецький М.В. Інтерполяційна функція Гауса як засіб мобільного аналізу даних/ М.В. Городецький, Ю. В. Сидоренко// Сучасні проблеми наукового забезпечення енергетики: Матеріали XVIII Міжнародної науково-практичної конференції молодих вчених та студентів, м. Київ, 23–26 квітня 2020 р. У 2 т. –К. : "КПІ ім. Ігоря Сікорського", 2020. –Т.2. – С.88

16. Гуриков С. Р. Введение в программирование на языке Visual C# — Форум, Инфра-М, 2013. - 448 с.

17. Дарвин, Я. Ф. Android сборник рецептов / Я. Ф. Дарвин. – М.: Вильямс, 2016. – 768 с.

18. Козлов, О. М. Системы автоматического регулирования: практикум по моделированию / О. М. Козлов, Б. Ф. Каташов, О. В. Карташов. – М.:Феникс, 2015. – 464 с.

19. Лук’яненко С. О. Интерполирование и апроксимация функций // Методические указания к выполнению лабораторних работ по курсу «Математические методы и модели в расчетах на ЭВМ». — К.: КПИ, 1990. — 100 с.

20. Лук'яненко С. О. Числові методи в інформатиці : навч. посіб. / С.О. Лук'яненко ; НТУУ "КПІ". - К. : НТУУ "КПІ", 2007. - 122 с.

21. Магда, Ю. І. Мікроконтролери серії 8051: практичний підхід / Ю. І. Магда. – М.:Академічний проект, 2008. – 837 с.

22. Мартин Р. С. Принципы, паттерны и методики гибкой разработки на языке C# — Символ-Плюс, 2011. - 768 с.

23. Рихтер Д. CLR via C#. Программирование на платформе Microsoft .NET Framework 2.0 на языке C# — Питер, 2007. — 656 с.

24. Робинсон С. C# для профессионалов — М.: Лори, 2005. — 396 с.

25. Самарский А.А. Введение в численные методы : учеб. пособ. для вузов / А. А. Самарский ; Московский государственный университет им. М. В. Ломоносова. - Москва : Лань, 2005. - 272 с.

26. Семакин И. Г. Основы программирования. / И. Г. Семакин, А. П. Шестаков. — М.: Мир, 2006. — 346 с.

27. Сидоренко Ю.В., Комп'ютерна реалізація різних способів параметризації інтерполяційної функції Гауса / Ю.В. Сидоренко, А.В. Сацкова // Прикладна геометрія та інженерна графіка — К.:КДТУБА, 2003, вип.72 — С.63-67.

28. Страуструп, Б. Язык программирования C++ / Б. Страуструп. – 2-е изд., доп. и перераб. – М.:Вильямс, 2002. – 369 с.

29. Турчак Л. И. Основы численных методов / Турчак Л. И., М.: Наука, 1987. — 320 с.

30. Фленов М. Библия C# — БХВ-Петербург, 2009. - 560 с.

31. Хилл Ф. OpenGL программирование компьютерной графики / Ф. Хилл — СПб.: Питер, 2006. — 1088 с.

32. Шилдт Г. Полный справочник по C#, 4-е издание. пер. с англ./ Герберт Шилдт. — М. : Издательский дом “Вильямс”, 2011. — 800 с.

33. Шилдт Г. Java 8. Руководство для начинающих, 6-е издание. пер. с англ./ Герберт Шилдт. — М. : Издательский дом “Вильямс”, 2015. — 712 с.

ДОДАТОК 1

Вплив коефіцієнта згладжування на вигляд інтерполяційної функції Гауса

Специфікація

КПІ ім. Ігоря Сікорського ТР6205_20Б

Аркушів 2

2020

Позначення	Найменування	Примітки
Документація		
КПІ ім. Ігоря Сікорського ТР6205_20Б 81-1	Записка.doc	Пояснювальна записка
Компоненти		
КПІ ім. Ігоря Сікорського ТР6205_20Б 12-1	Visualization.cs	Модуль візуалізації
КПІ ім. Ігоря Сікорського ТР6205_20Б 12-2	Counting.cs	Модуль обчислень
КПІ ім. Ігоря Сікорського ТР6205_20Б 12-3	Covid19.cs	Модуль взаємодії з даними по захворюванню COVID-19
КПІ ім. Ігоря Сікорського ТР6205_20Б 12-4	Report.cs	модуль побудови звіту
КПІ ім. Ігоря Сікорського ТР6205_20Б 13-2	Опис.docx	Опис модуля обчислень

ДОДАТОК 2

Вплив коефіцієнта згладжування на вигляд інтерполяційної функції Гауса

Модуль обчислень

Текст програмного модуля

КПІ ім. Ігоря Сікорського ТР6205_20Б 12-2

Аркушів 10

2020

```
using SystemSolver;
using System;
using ZedGraph;

namespace Interpolation {
        public class GaussianInterpolation : InterpolationBase {
      private readonly double[] basis;

      public GaussianInterpolation(PointPairList inputPoints, double alpha) :
base(inputPoints)
      {
         Alpha = alpha;
         basis = findGaussianBasis();
      }

      public override string Name { get; protected set; } = "Gaussian Non Parametric";

      public override PointPair GetPoint(double Xl)
      {
         double G = 0;

         for (int i = 0; i < InputPoints.Count; i++) {
            double Xr = InputPoints[i].X;
            G += basis[i] * Math.Exp(-Alpha * Math.Pow(Xl - Xr, 2));
         }

         return new PointPair() { X = Xl, Y = G };
      }

      public virtual double Alpha { get; protected set; }

      private double[] findGaussianBasis()
      {
         int n = InputPoints.Count;

         double[,] Ab = new double[n, n + 1]; // n + 1 because the result (Y) will be here

         // Create Basis matrix. Xl -> Xn, Xr -> X[1..n] in formula
         for (int i = 0; i < n; i++) {
            double xL = InputPoints[i].X;

            for (int j = 0; j < n; j++) {
               double xR = InputPoints[j].X;
```

```
                Ab[i, j] = Math.Exp(-Alpha * Math.Pow(xL - xR, 2));
            }

            Ab[i, n] = InputPoints[i].Y;
        }

        try {
            return GaussJordanElimination.SolveSystem(Ab, n);
        } catch (NoSolutionException e) {
            throw new InterpolationException("Решения для базисов ф-и Гаусса НЕ
найдено!", e);
        } catch (InfiniteSolutionException e) {
            throw new InterpolationException("Решения для базисов ф-и Гаусса
содержит бесконечность!", e);
        } catch (Exception e) {
            throw new InterpolationException("GaussJordanElimination Error", e);
        }
    }
  }

  }


using System;
using System.Linq;
using ZedGraph;

namespace Interpolation {
      public class GaussianParametricInterpolation : GaussianInterpolation {
    private GaussianInterpolation gaussianXt;
    private GaussianInterpolation gaussianYt;

    public GaussianParametricInterpolation(PointPairList inputPoints, double alpha,
ParametricType type) : base(inputPoints, alpha)
    {
        Type = type;
        Name += type.ToString();

        XTArray = new PointPairList();
        YTArray = new PointPairList();

        if (type == ParametricType.Normal) {
            for (int i = 0; i < inputPoints.Count; i++) {
```

```
            XTArray.Add(new PointPair(i, inputPoints[i].X)); // fill Xt(t)
            YTArray.Add(new PointPair(i, inputPoints[i].Y)); // fill Yt(t)
        }

    } else if (type == ParametricType.Summary) {
        double previousT = 0;
        XTArray.Add(new PointPair(previousT, inputPoints[0].X)); // fill Xt[0](t)
        YTArray.Add(new PointPair(previousT, inputPoints[0].Y)); // fill Y[0](t)

        for (int i = 1; i < inputPoints.Count; i++) {
            PointPair prevPoint = inputPoints[i - 1];
            PointPair currPoint = inputPoints[i];
            previousT += distanceBetween(prevPoint, currPoint);

            XTArray.Add(new PointPair(previousT, inputPoints[i].X)); // fill Xt(t)
            YTArray.Add(new PointPair(previousT, inputPoints[i].Y)); // fill Y(t)
        }
    }

    TMin = XTArray.First().X;
    TMax = XTArray.Last().X;
}

public override string Name { get; protected set; } = "Gaussian Parametric ";

public ParametricType Type { get; protected set; }

public new double Alpha { get => base.Alpha; set => base.Alpha = value; }

public double GetT(int index) => XTArray[index].X /*T*/;

public double TMax { get; protected set; }

public double TMin { get; protected set; }

public PointPairList XTArray { get; protected set; }

public PointPairList YTArray { get; protected set; }

public override PointPair GetPoint(double T)
{
    gaussianXt = gaussianXt ?? new GaussianInterpolation(XTArray, Alpha);
    gaussianYt = gaussianYt ?? new GaussianInterpolation(YTArray, Alpha);
```

```
            return new PointPair(gaussianXt.GetPoint(T).Y, gaussianYt.GetPoint(T).Y);
        }

        private double distanceBetween(PointPair a, PointPair b)
        {
            double x1 = a.X, x2 = b.X,
                   y1 = a.Y, y2 = b.Y;
            return Math.Sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1));
        }
    }

}
```

```
using ZedGraph;

namespace Interpolation {
    public class LagrangeInterpolation : InterpolationBase {

        public LagrangeInterpolation(PointPairList inputPoints) : base(inputPoints)
        { }

        public override string Name { get; protected set; } = "Lagrange";

        public override PointPair GetPoint(double X)
        {
            double L = 0;

            for (int i = 0; i < InputPoints.Count; ++i) {
                double l = 1;

                for (int j = 0; j < InputPoints.Count; ++j)
                    if (i != j)
```

```
                l *= (X - InputPoints[j].X) / (InputPoints[i].X - InputPoints[j].X);

            L += InputPoints[i].Y * l;
        }

        return new PointPair(){ X = X, Y = L };
    }
  }
using System;
using System.Globalization;
using System.Text;

namespace SystemSolver {

  public static class GaussJordanElimination {
    private static readonly string logFormatter = "F3";

    public static readonly bool LogIsOn = false;

    // a - matrix Ab
    // n - order of Matrix(n)
    /*
      Example b:
      double a[,] = {{ 0, 2, 1, 4 },
                     { 1, 1, 2, 6 },
                     { 2, 1, 1, 7 }};
      b = 4, 6, 7
     */
    public static double[] SolveSystem(double[,] a, int n)
```

```
{
    // Performing Matrix transformation
    int flag = performOperation(a, n);

    if (flag == 1)
        flag = checkConsistency(a, n, flag);

    // Printing Final Matrix
    writeLine("Final Augmented Matrix is: ");
    printMatrix(a, n);

    // Return Solutions(if exist)
    return getSolution(a, n, flag);
}

// function to reduce matrix to reduced
// row echelon form.
private static int performOperation(double[,] a, int n)
{
    int i, j, c, flag = 0;

    // Performing elementary operations
    for (i = 0; i < n; i++) {
        int k;
        if (a[i, i] == 0) {
            c = 1;
            while ((i + c) < n && a[i + c, i] == 0) {
                c++;
                writeLine("i:" + i + " c:" + c + " i+c:" + (i + c));
```

```
            }
            if ((i + c) == n) {
                flag = 1;
                break;
            }
            for (j = i, k = 0; k <= n; k++) {
                double temp = a[j, k];
                a[j, k] = a[j + c, k];
                a[j + c, k] = temp;
            }
        }

        for (j = 0; j < n; j++) {

            // Excluding all i == j
            if (i != j) {

                // Converting Matrix to reduced row
                // echelon form(diagonal matrix)
                double p = a[j, i] / a[i, i];

                for (k = 0; k <= n; k++)
                    a[j, k] = a[j, k] - (a[i, k]) * p;
            }
        }
    }
    return flag;
}
```

```
// Function to get the desired result
// if unique solutions exists, otherwise
// prints no solution or infinite solutions
// depending upon the input given.
private static double[] getSolution(double[,] a, int n, int flag)
{
    if (flag == 2) {
        throw new InfiniteSolutionException();
    } else if (flag == 3) {
        throw new NoSolutionException();
    }

    double[] solution = new double[n];
    StringBuilder line = new StringBuilder("Result is: ");

    // Getting the solution by dividing constants by
    // their respective diagonal elements
    for (int i = 0; i < n; i++) {
        solution[i] = a[i, n] / a[i, i];
        line.Append(solution[i].ToString(logFormatter)).Append(" ");
    }

    writeLine(line.ToString());

    return solution;
}

// To check whether infinite solutions
// exists or no solution exists
```

```
private static int checkConsistency(double[,] a, int n, int flag)
{
    int i, j;
    double sum;

    // flag == 2 for infinite solution
    // flag == 3 for No solution
    flag = 3;
    for (i = 0; i < n; i++) {
        sum = 0;
        for (j = 0; j < n; j++)
            sum += a[i, j];
        if (sum == a[i, j])
            flag = 2;
    }
    return flag;
}
// Function to print the matrix
private static void printMatrix(double[,] a, int n)
{
    for (int i = 0; i < n; i++) {
        StringBuilder line = new StringBuilder();
        for (int j = 0; j <= n; j++)
            line.Append(a[i, j].ToString(logFormatter)).Append(" ");
        writeLine(line.ToString());
    }
}
```

ДОДАТОК 3

Вплив коефіцієнта згладжування на вигляд інтерполяційної функції Гауса

Модуль обчислень

Опис програмного модуля

КПІ ім. Ігоря Сікорського ТР6205_20Б 13-2

Аркушів 7

2020

АНОТАЦІЯ

Розроблений програмний модуль обчислень призначений для розрахунки точок інтерполяції методами класичної Гаус-функції, параметричної Гаус-функції сумарної Гаус-функції та методом Лагранжа; також модуль реалізовує набір функцій, над якими проводиться дослідження варіаційної змінної a.

Як вхідні дані модуль отримує набір функцій для інтерполяції, величини кроку інтерполяції та тип кроку.

Як результат модуль повертає масив точок інтерполяції.

Мова програмної реалізації — C#.

Середовище розробки — Visual Studio 2020.

ЗМІСТ

1. ЗАГАЛЬНІ ВІДОМОСТІ

Назва програмного модуля — “ Модуль обчислень ”. Мова програмної реалізації — C#. Середовище розробки — Visual Studio 2020.

Модуль “Модуль обчислень” є складовою частиною програмного продукту, що обчислює значення інтерполяції методами Лагранжа та методами Гаус-функцій.

Реалізація модулю представлена чотирма класами. Цей модуль має єдиний інтерфейс для взаємодії для всіх методів інтерполяції, що спрощує використання його в інших модулях програмного продукту. Для кожного методу інтерполяції необхідно передати як параметри змінну α, тип кроку та мінімум і максимум по осі координат x.

2. ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ МОДУЛЯ

Функціональним призначенням модуля “Модуль обчислень” є розв’язання задачі інтерполяції для будь-якого каркасу точок такими методами, як: Лагранжа, Гаус звичайний, Гаус-параметричний, Гаус-сумарний.

Модуль за допомогою звернення до інших методів знаходить розв’язання поставленої задачі. Він викликає виконання таких методів:

— getPointAt — повертає точку інтерполяції на вказаному в параметрі кроку;

— State класу State — будує матрицю для порівняння;

— Compare класу Comparer — виконує порівняння попередньої й наступної точки;

— getCurrentStep — повертає величину кроку інтерполяції на поточній ітерації;

— getAlgorithmScore — нормалізує набір проінтерпольованих точок та розраховує оцінку похибки алгоритму інтерполяції.

3. ВЗАЄМОДІЯ МОДУЛЯ З ІНШИМИ КОМПОНЕНТАМИ СИСТЕМИ

Модуль “Модуль обчислень” взаємодіє з модулем “Модуль візуалізації” через свій інтерфейс. Цей модуль є повністю незалежним та використовується модулем візуалізації для побудови інтерполяції кожної з функцій по черзі.

Модуль працює у фоновому процесі, тож для взаємодії з ним немає потреби припиняти обробку вводу та виводу до користувача.

При закінченні інтерполяції з заданими параметрами модуль створює повідомлення для інших модулів про те, що робота закінчена і можна отримати результати.

4. ВХІДНІ ТА ВИХІДНІ ДАНІ

Як вхідні дані модуль отримує набір функцій для інтерполяції, величини кроку інтерполяції та тип кроку, а також — мінімальний та максимальне значення *x*.

Як результат модуль повертає масив з точками відновленої функції, а також оцінку алгоритмів інтерполяції.

ДОДАТОК 4

Вплив коефіцієнта згладжування на вигляд інтерполяційної функції Гауса

Апробації

КПІ ім. Ігоря Сікорського ТР6205_20Б

Аркушів 21

2020

ДОДАТОК 5

Вплив коефіцієнта згладжування на вигляд інтерполяційної функції Гауса

Акт впровадження

КПІ ім. Ігоря Сікорського ТР6205_20Б

Аркушів 2

2020

ЗАТВЕРДЖУЮ
Керівник підприємства

В.В. Сікачина

АКТ ВПРОВАДЖЕННЯ

результатів дипломної роботи Городецького М.В. на тему «Вплив коефіцієнта згладжування на вигляд інтерполяційної функції Гауса», яка виконана в Національному технічному університеті України «Київський політехнічний інститут ім. Ігоря Сікорського» («КПІ ім. Ігоря Сікорського»)

”17” травня 2020 р.

Нами, представниками кафедри автоматизації проектування енергетичних процесів і систем «КПІ ім. Ігоря Сікорського» та ТОВ «БІ-ХАБ», даний акт складено про те, що для використання в розробках спеціалізованого програмного забезпечення ТОВ «БІ-ХАБ» прийняті результати дипломної роботи Городецького М.В., а саме: програмний засіб “Вплив коефіцієнта згладжування на вигляд інтерполяційної функції Гауса”, реалізований за допомогою інтерполяцій на основі функції Гауса у вигляді запису на CD, а також документацію програмного супроводу.

Представник кафедри АПЕПС
«КПІ ім. Ігоря Сікорського»
Керівник дипломної роботи Сидоренко Юлія Всеволодівна _____
(прізвище, ім’я, по батькові) (підпис)

Представник ТОВ «БІ-ХАБ»
Керівник підприємства Сікачина Віталій Вікторович _____
(прізвище, ім’я, по батькові) (підпис)